# Practical Reference Manual

## X1 リファレンスノゥト

杉浦勇一, 難波生, 仲谷和人, 松村守 共著

# N1 リファレンスノート

杉浦勇一, 難波生, 仲谷和人, 松村守 共著

# まえがき

X１シリーズは，その高い性能のため多くのユーザーを得ていますが，解説書籍が比較的少ないうえ，マニュアルもけっして十分とはいえず，もの足りない思いをされた方も多いことでしょう。前書にあたる『X１マシン語プログラミング入門』でも「マシン語入門」という性格上，X１のハードウェア機能についてはあまり触れていません。本書はこれを補う意味で，HuBASICの内部構造とハードウェア機能の使い方，応用について多くのページをさいて解説しています。

本書では，CRTCやPSGなどかなりハード的な解説があり，わかりにくい点があるかも知れませんが，多くのサンプル・プログラムを掲載していますので，実験を繰り返し，体験的に覚えることができます。また，それがコンピューターを知るうえで最も効果的であり，つまり近道であるといえます。

本書は全ページにわたって資料集といえますが，特に付録のIOCS，I/Oポート，サブCPUコマンド表はクイック・リファレンスとして役に立つでしょう。

本書がまさに“もう一冊のマニュアル”として皆さんの参考になれば幸いです。

1985年 4月 著者

## ■ 本書を読む前に ■

本書を執筆するにあたって，使用したものは次のとおりです。

(1) コンピュータ…CZ-800C，シリアル NO.312373

(2) プリンタ…PR-201(NEC 製)

(3) アセンブラ…『マシン語プログラミング入門』に記載されているエディタ・アセンブラ

第2章でメモリ・ダンプを数多く掲載していますが，プリンタが NEC 製であるため，一部 ASCII コードが違うものがあります。ASCII コードの部分は本文とは関係ないものなので特に気にする必要はありません。また，アセンブラは標準的なもので，他のもので代用できるでしょう。

# CONTENTS

# 第3章　画面構成——79



# 第4章　周辺チップ——125

# 第1章
## システム概要



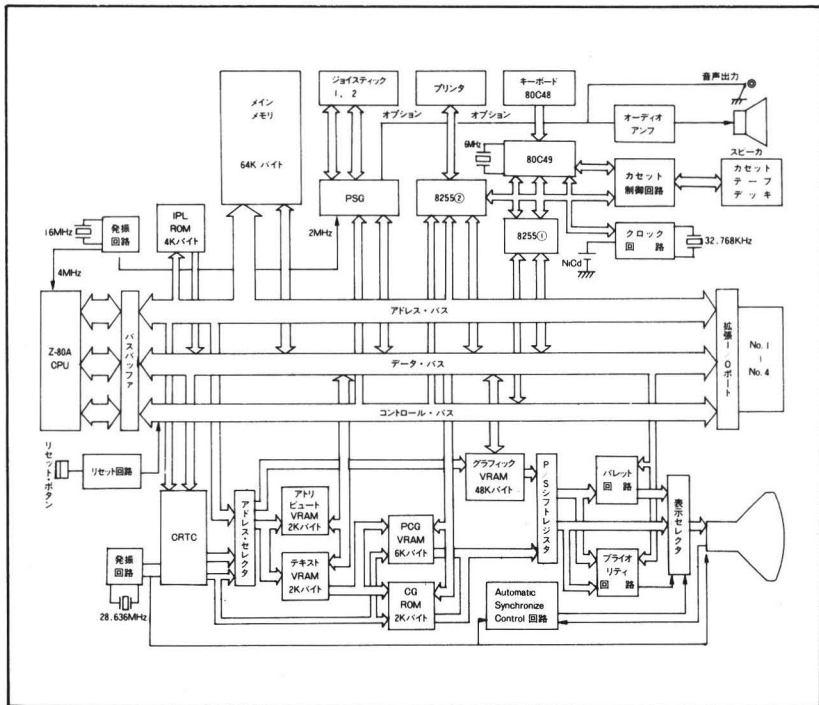
テクニカルな解説は次章以降に任せることにして本章は導入編として，X1の機能概説と特徴ある I/Oポートについて解説していきます。すでに予備知識のある方は読みとばして結構です。 また， X1 Turboの拡張された部分についても触れます。

# 1-1 ブロック図

**図1-1**にX1のブロック図を示します。一番最初に発売されたCZ-800Cの図ではグラフィックRAMや拡張I/Oポートがオプションになっています。各ブロックを簡単に説明します。

### ●IPL ROM

X1は，電源を入れるとまずIPL (Initial Program loader) が動作します。IPL ROMの大きさは4Kバイトで，この中には，ディスク入力ルーチン，カセット入力ルーチン，ROM入力ルーチン，タイマーセット・ルーチンが含まれています。

図1-1 ブロック図

**●メイン・メモリ**

Z80のI/O空間いっぱいの64Kバイトの RAM を備えています。BASIC などのシステム・ソフトウェアはもちろんのこと，BASICのテキストなども格納されてます。

**●テキスト VRAM**

テキストの表示用に2Kバイト持ち，80字×25行を1画面または40字×25行を2画面分持つことができます。また，テキストVRAMに1対1に対応して，表示文字の色，モード(反転，点滅)，大きさ，PCGかCGの選択を設定するアトリビュート（属性）VRAMを2Kバイト分持っています。

**●グラフィック VRAM**

48Kバイト持っており，640×200ドットの解像度で，ドット単位に色指定ができます。単色であれば3画面分持てます。また，320×200ドットのモードがあり，この場合カラーだと2画面，単色だと6画面持てます。

**● CG ROM**

テキスト画面に表示する文字のパターン・データが入っているROMです。256文字分で1文字8バイトですからROMの容量は2Kバイトとなります。CGはキャラクタ・ジェネレータの略です。

**● PCG**

PCGはプログラマブル・キャラクタ・ジェネレータの略でX1のユニークな機能のひとつです。256文字分定義でき，ドット単位に色指定できます。

**● CRTC**

画面表示を制御するLSI(CRTコントローラ)です。X1ではポピュラーなHD46505-SP（日立製）を使っています。

**●パレット回路**

TTLによって構成された回路でグラフィック画面に書かれたものを瞬時に他の色へ変えることができます。テキスト画面に書かれたものは色が変わりません。

●プライオリティ回路

テキスト画面とグラフィック画面の優先順位を設定するものです。X1ではグラフィック画面の色別に設定できるので，たとえばテキスト画面をグラフィック画面の青，黄色の陰にするといったことができます。

● PSG

プログラマブル・サウンド・ジェネレータの略で，プログラムによって様々な音を発生できます。AY-3-8910という1チップのLSIによってコントロールされます。また，このLSIは2つのI/Oポートを持ち，X1ではジョイスティックの入力に使われています。

● 8255 (PPI)

X1では2個の8255を使っています。1つはZ80が管理しプリンタなどのコントロールに，もう1つはサブCPU(80C49)が管理しカセットなどのコントロールに使われています。

● 80C49，80C48

X1ではメインCPUの負担を軽減するため，2個のサブCPUを使っています。80C48はキー入力，80C49はカセットのコントロールなどに使われています。

# 1-2 I/Oポート

Z80では，一般的に周辺機器の入出力をI/Oポートを介して行います。入出力はZ80のIN，OUT命令を使って，周辺機器とデータのやりとりを行っています。どのポートがどういう役割か，ということはシステムによって決まっています。

Z80では，I/Oポートの空間は0～255までの256バイトで，IN,OUT命令でもポート番号の指定は8ビットで行います。これらの命令には，

① IN　　A,　(nn)

② OUT　(nn)，A

③ IN　　8ビット・レジスタ，(C)

④ OUT　(C)，8ビット・レジスタ

などがあります。ここで，nnは8ビットの数値，8ビット・レジスタは，A，H，L，B，C，D，Eのレジスタです。こう書くとZ80は256バイトまでしかI/O空間を持てないように思いますが，X1では64KバイトのI/O空間を持っています。

これは，別に特別なZ80を使っているわけではなく，Z80はもともと64KバイトのI/Oの空間を持てるのです。64Kバイトの空間をアクセスするためには16ビット長で指定しなければなりません。①,②の命令はAレジスタの内容をnnで示されるI/Oポートに入出力を行うものですが，実はアドレス・バスにはAレジスタを上位，nnを下位にしたデータが出力されているのです。たとえば，Aレジスタに12Hを入れ，

```
OUT　(34H)，A
```

を実行すると，I/Oアドレス1234Hに12Hが出力されます。また，③，④の場合はBレジスタがI/Oアドレスの上位8ビットとして出力されます。たとえば，

```
LD    BC, 1234H
IN    B, (C)
```

といった場合でも，I/O アドレス 1234H から B レジスタへデータが入力されます。X1 やソニーの SMC シリーズでは，こういった Z80 の性質を利用してグラフィック RAM のように大きな空間も占めるものを I/O 空間に割り当てています。

Z80 にはこのほかに INIR や OTDR といったブロック出力命令がありますが，これは B レジスタをカウンタとして使うために X1 では，まず使う機会はないでしょう。

さて，X1 で入出力を行うときに，常に16ビットすべてを指定しなければならないわけではありません。周辺デバイスによっては，上位 8 ビットしかデコードしていないものもあるからです。この場合下位 8 ビットはどんは値でもよいことになります。巻末付録に I/O マップ（I/O ポートがどう割り当てられているかを示す図）がありますが，このなかで'＊＊'と書いてあるのは，デコードされずどんな値でもよいことを示しています。

# 1-3 X1 Turbo

X1シリーズの最上位機種として発表されたX1Turboはこれまでのマイナーチェンジと違って，コンパチビリティを保ちながら大きな拡張が施されました。拡張された点は次のとおりです。

**❶グラフィック**

グラフィックVRAMをもう1バンク(1部オプション)設け，計96Kバイトになりました。ことため400ラインの高解像度が使え，表現力がさらにアップしました。また，画面モードが多彩になりました。

**❷漢字 VRAM**

漢字VRAMが搭載されたことによって，漢字表示を普通のテキスト表示と同じように手軽に，なおかつ高速に行うことができます。

**❸周辺チップ**

周辺チップにZ80ファミリであるDMA，SIO，CTCが追加されました。DMAはフロッピーディスクなどの高速入出力に，SIOはRS-232C，マウスとの入出力に，CTCはRS-232Cのボーレート設定，タイマ割り込みに使われています。

**❹ソフトウェア**

BASICがさらに拡張されましたが，BIOS ROMを32Kバイト持ち，フリーエリアは逆に増えています。

この他，細かな点でいろいろと拡張されていますが，完全に上位コンパチになっています。そのため本書の内容はそのままX1Turboにも適用できます。巻末付録にはX1TurboのI/Oポートなども掲載しています。

# 第2章

# HuBASIC



HuBASICは非常に機能の高いBASICで,X1のユニークなハードウェア機能を十分に活かせる言語です。このHuBASICをマシン語プログラムとして見た場合,各所でいろいろな工夫が見られ,大変興味深いものです。本章では,マシン語レベルからHuBASICを見直し,マニュアルには載っていない使い方,機能を拡張する方法などを紹介します。

なお,説明はテープ・バージョン(CZ8CB01)を中心に行っていき,ディスク・バージョン(CZ8FB01)との違いがある場合は,適時説明を入れることにします。

# 2-1 HuBASICの特徴

X1のHuBASICの主な特徴としては，次のようなものが挙げられます。
①プライオリティ，パレット，論理座標系を含む強力なグラフィック・コマンド。
②スーパーインポーズ，チャンネル，音量，TVタイマー，スムーズ・スクロールなどのTVコントロール。
③ドット単位で色指定できるPCGと豊富なキャラクタ表示命令。
④2700ボーの高速SAVE，LOADとAPSS（プログラムの頭出し）などのカセット・コントロール機能。
⑤省略形，SEARCH,AUTO命令による強力なエディタ機能。
⑥ラベル命令や豊富なループ命令によるプログラムの構造化が可能。
⑦単精度9桁，倍精度で16桁の高精度演算。SINなどの関数でも倍精度演算が可能で，しかも2進・8進定数も扱うことができる。

などなど，良い点をあげればきりがありません。なにしろ命令数が合計で200種類以上あるのですからX1のハードウェアの性能を引き出すのに十分な言語と言えます。

ただし，それだけ多くの命令を持たせたためフリーエリアが24Kバイト（カセットの場合）と少なめになっています。X1 Turboではこの問題は解決されていて，最大で80Kバイト近くのフリーエリアが得られます。

他のパソコンではマイクロソフト系のBASICが採用されていますが，HuBASICの文法もマイクロソフト系のそれと同じ系列のBASICだといえます。そのため，他のパソコン用のプログラムから移植するときも比較的少ない手間ですみます。

さらに，SHARP系のBASICのコマンドをも包含しているため,MZ系を使い慣れた人でもX1に移行しやすいようです。

# 2-2 メモリ・マップ

HuBASICのメモリ・マップを**図2-1**に示します。カッコ内の値はディスクBASICのアドレスです。ただし，テープでもディスク・バージョンでも同一アドレスのときは，一方を省略しています。このメモリ・マップでは各エリアの占める大きさがわかりやすいように，できるだけメモリ容量に忠実に書いてみました。

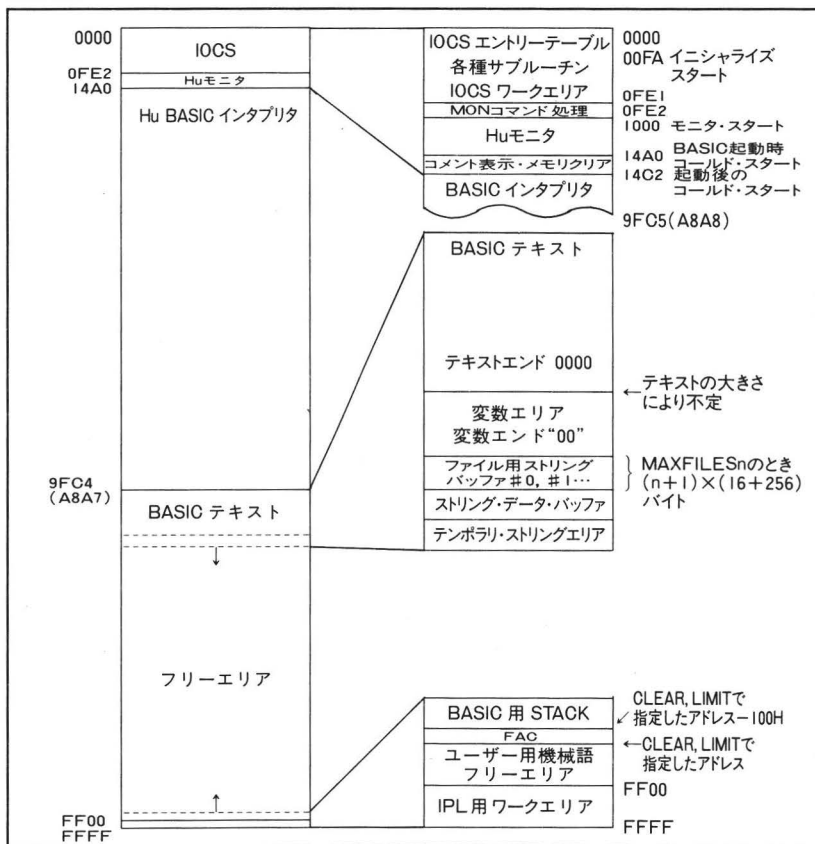
**図2-1 メモリ・マップ**

IOCS，インタープリタ部分およびFF00H以降のIPLなどのワーク・エリアはマップに示されるままですが，テキスト・エリアや変数エリアは状況によって変化します。それぞれの先頭アドレスは特定のワーク・エリアに入っています。このワーク・エリアのアドレスとBASIC起動時の初期値を**表2-1**に示します。

**表2-1 ワーク・エリアと初期値**

| 内容 | 格納アドレス カッコ内はDISKBASIC | 初期値 | | 備考 |
|---|---|---|---|---|
| | | BASIC起動前 | 起動後 | |
| BASICテキスト・トップ | 9D52 (A635) | 9FC5 | 9FC5 (A8A8) | NEW ON命令で変えられる |
| 変数エリア・トップ | 9D46 (A629) | A0EF | 9FC7 (A8AA) | VARPTR命令はこのエリア内のアドレスを示す |
| ファイル用ストリング・バッファトップ | 9D48 (A62B) | A0FC | 9FC8 (A8AB) | STRPTR命令で値を見ることができる |
| ストリング・データ・バッファトップ | 35F6 (3628) | A31C | A1E8 (ABDB) | |
| テンポラリ・ストリングエリア・トップ | 9D4A (A62D) | A3D6 | A1EA (ABDD) | |
| フリーエリア・トップ | 35EF (3621) | A3D6 | A1EA (ABDD) | |
| FACトップ | 9D4C (A62F) | FD00 | FE00 (FE00) | CLEAR, LIMITのアドレス－100Hである |
| ユーザー用機械語フリーエリアトップ | 9D4E (A631) | FE00 | FF00 (FF00) | CLEAR, LIMIT命令で変えることができる |
| IPL用ワークエリアトップ | 9D50 (A633) | FF00 | FF00 (FF00) | 半固定 |

次に各エリアについて簡単に説明しておきます。

● IOCS (Input Output Control System)はその名称のとおり，ハードウェアに直接関係するデータの入出力をつかさどる部分です。

●テキストの先頭アドレスはNEW ON命令によって，変えることができます。終了アドレスはテキストの長さによって変わります。

●変数エリアは数値変数の種類，変数名と値そのものが入ります。文字変数の場合は種類，変数名，文字列の長さと文字列が実際に格納されているアドレスが入ります。

●ファイル用ストリング・バッファは周辺装置とデータをやりとりするときの一時的なバッファ256バイトとデータの入出力先や処理アドレスの入った部分16バイトからなっていま

す。MAXFILES 命令で1度に扱えるファイルが指定できるのでファイルの数によってこのエリアの大きさが決定されます。MAXFILES n のときこのエリアは(n+1)×(16+256)バイトとなります。テープ BASIC の場合，起動時にファイルの数は#0，#1の2個，ディスク BASIC の場合は3個になっています。

このうち，#0のファイルはシステム用で，カセットへの入出力やリストなどを出力するときに使われます。

● ストリング・データ・バッファは文字変数の実際の文字列データが入るエリアです。

● テンポラリ・ストリング・バッファは文字変数の多重処理，ファイルからの入力時に一時的に使われるエリアです。

● BASIC 用スタックは GOSUB 文の戻り番地などを入れておくエリアです。

● FAC (Floating ACcumulator)は浮動小数点演算用のアキュムレータです。X1 では100H (256バイト) と決められています。浮動小数点の代入，演算や USR 命令などでわたされるデータは一時 FAC に転送され，FAC を経由して処理されます。FAC の先頭のアドレスは CLEAR や LIMIT 命令で決定されます。初期値は FF00Hです。

● FF00H～FFFFH はマニュアルでは IPL 用のワーク・エリアとなっていますが，実際はモニタでも BASIC でも使われる汎用のワーク・エリアです。 FF00H からはキー入力バッファやファイルのインフォメーション・バッファとして，FFFFH の方からはスタック用として使われます。

# 2-3 中間コード

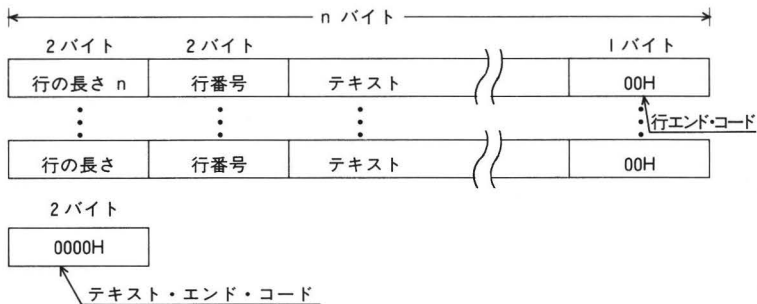
BASICのプログラムは9FC5H (ディスクの場合はA8A8H) から格納されています。**図2-2**にLISTコマンドで出力したリストとモニタのDのコマンドでダンプしたリストを示します。これを見てもわかるようにLISTで見えるそのままの形でメモリに格納されているわけではなく，できるだけメモリを使わないように工夫されています (**図2-3**)。

**図2-2 テキストの格納**

```
10 REM SUMLPE
20 A=2:B=3
30 PRINT A+B
40 END

:9FC5=0D 00 0A 00 97 20 53 55 /....|SU
:9FCD=4D 4C 50 45 00 0C 00 14 /MLPE....
:9FD5=00 41 F4 03 3A 42 F4 04 /.A日.:B日.
:9FDD=00 0A 00 1E 00 8F 20 41 /.....+ A
:9FE5=F7 42 00 06 00 28 00 98 /秒B...(.r
:9FED=00 00 00 00 00 4F 39 5D /.....O9]
```

注：ディスクではA8A8Hから。

**図2-3 テキストの格納形式**

注：行番号および行の長さは下位，上位の順に格納されています。

行番号の最小値は1，最大値は65534(FFFEH)となっていますが，メモリを直接操作することで0行や65535行をつくることも可能です。実行しても正常に動きます。ただし，その行はBASICのエディタでは書き換えることはできなくなります。例を**図2-4**に示します。

図2-4 **テキストの格納**

```
10 ' SUMPLE
20 '  LINE 0 & 65535
30 PRINT "ABC"
40 PRINT "ABC"

:9FC5=0E 00 0A 00 3A 27 20 53 /....:' S
:9FCD=55 4D 50 4C 45 00 17 00 /UMPLE...
:9FD5=14 00 3A 27 20 20 4C 49 /..:'  LI
:9FDD=4E 45 20 30 20 26 20 36 /NE 0 & 6
:9FE5=35 35 33 35 00 0C 00 1E /5535....
:9FED=00 8F 20 22 41 42 43 22 /.+  0
:9FF5=00 0C 00 28 00 8F 20 22 /...(.+
:9FFD=41 42 43 22 00 00 00 00 /ABC"....

:9FC5=0E 00 00 00 3A 27 20 53 /....:' S
:9FCD=55 4D 50 4C 45 00 17 00 /UMPLE...
:9FD5=00 00 3A 27 20 20 4C 49 /..:'  LI
:9FDD=4E 45 20 30 20 26 20 36 /NE 0 & 6
:9FE5=35 35 33 35 00 0C 00 1E /5535....
:9FED=00 8F 20 22 41 42 43 22 /.+  0
:9FF5=00 0C 00 FF FF 8F 20 22 /...   +
:9FFD=41 42 43 22 00 00 00 00 /ABC"....

0 ' SUMLPE
0 '  LINE 0 & 65535
30 PRINT "ABC"
65535 PRINT "ABC"
```

また，1行の長さも255文字と決められていますが，行の長さは2バイトで表されています。ですからこれもメモリを直接操作することで長い行をつくることができます。もっともこれはあまり意味のあることではありませんが，PLAY 命令などでどうしても1行に入れたいときやリストを隠したいときには有効でしょう。例を**図2-5**に示します。10行目の行の長さ

のところには,

| 009BH(10行の長さ)+00FFH(20行の長さ)−1=0109H |
|---|

を入れます。さらに10行の終わりの印である00H, 20行の長さ, 行番号のあったところはスペース(20H)かコロン(3AH)を書き込んでおきます。変更後, リストをとってみても255文字しか表示されませんが, きちんと最後まで実行できます。

**図2-5 255文字より長い行をつくる**

```
10 TEMPO 7500
20 PLAY "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAB"


 :9FC5=0B 00 0A 00 FE A3 20 12 /.... ｣ .
 :9FCD=4C 1D 00 FF 00 14 00 FE /L.. ...
 :9FD5=8B 20 22 41 41 41 41 41 /▮  0
 :9FDD=41 41 41 41 41 41 41 41 /AAAAAAAA
 :9FE5=41 41 41 41 41 41 41 41 /AAAAAAAA
 :9FED=41 41 41 41 41 41 41 41 /AAAAAAAA
 :9FF5=41 41 41 41 41 41 41 41 /AAAAAAAA
 :9FFD=41 41 41 41 41 41 41 41 /AAAAAAAA
 :A005=41 41 41 41 41 41 41 41 /AAAAAAAA
 :A00D=41 41 41 41 41 41 41 41 /AAAAAAAA
 :A015=41 41 41 41 41 41 41 41 /AAAAAAAA
 :A01D=41 41 41 41 41 41 41 41 /AAAAAAAA
 :A025=41 41 41 41 41 41 41 41 /AAAAAAAA
 :A02D=41 41 41 41 41 41 41 41 /AAAAAAAA
 :A035=41 41 41 41 41 41 41 41 /AAAAAAAA
 :A03D=41 41 41 41 41 41 41 41 /AAAAAAAA


 :9FC5=09 01 0A 00 FE A3 20 12 /.... ｣ .
 :9FCD=4C 1D 3A 3A 3A 3A 3A FE /L.:::::
 :9FD5=8B 20 22 41 41 41 41 41 /▮  0
 :9FDD=41 41 41 41 41 41 41 41 /AAAAAAAA
 :9FE5=41 41 41 41 41 41 41 41 /AAAAAAAA
 :9FED=41 41 41 41 41 41 41 41 /AAAAAAAA
 :9FF5=41 41 41 41 41 41 41 41 /AAAAAAAA
 :9FFD=41 41 41 41 41 41 41 41 /AAAAAAAA
 :A005=41 41 41 41 41 41 41 41 /AAAAAAAA
 :A00D=41 41 41 41 41 41 41 41 /AAAAAAAA
 :A015=41 41 41 41 41 41 41 41 /AAAAAAAA
 :A01D=41 41 41 41 41 41 41 41 /AAAAAAAA
 :A025=41 41 41 41 41 41 41 41 /AAAAAAAA
 :A02D=41 41 41 41 41 41 41 41 /AAAAAAAA
 :A035=41 41 41 41 41 41 41 41 /AAAAAAAA
 :A03D=41 41 41 41 41 41 41 41 /AAAAAAAA


10 TEMPO 7500:::::PLAY "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
```

さて，**図2-2**では“REM”や“PRINT”などの命令や“＋”などの演算子がそのままのASCIIコードで格納されていません。これらの命令語ははじめからBASIC用に予約されていて，変数名の先頭に使うことはできず，「予約語」と呼ばれています。

すべての予約語は中間コードの順にテーブルとして格納されています。この予約語はテープの場合28F6Hから2CDDHにあり，ディスクでは2921Hから2D0BHにあります。**図2-6**に示すようにASCIIコードで格納されていて1語の終わりの文字は最上位ビットをたてて区別しています。たとえば“GOTO”は次のようになります。

| アドレス | データ | | | | |
|---|---|---|---|---|---|
| 28F6H | 47H | 4FH | 54H | CFH←1語の終わりなので | |
| | G | O | T | O | 最上位ビットを立てる |

予約語の割り当てられていない中間コードのところには80Hが置かれています。

**図2-6 予約語（一部）**

```
:28F6=47 4F 54 CF 47 4F 53 55 /GOTﾏGOSU
:28FE=C2 47 CF 52 55 CE 52 45 /ﾂGﾏRUﾎRE
:2906=54 55 52 CE 52 45 53 54 /TURﾎREST
:290E=4F 52 C5 52 45 53 55 4D /ORﾅRESUM
:2916=C5 4C 49 53 D4 4C 4C 49 /ﾅLISﾔLLI
:291E=53 D4 44 45 4C 45 54 C5 /SﾔDELETﾅ
:2926=52 45 4E 55 CD 41 55 54 /RENUﾍAUT
:292E=CF 45 44 49 D4 46 4F D2 /ﾏEDIﾔFOﾒ
:2936=4E 45 58 D4 50 52 49 4E /NEXﾔPRIN
:293E=D4 4C 50 52 49 4E D4 49 /ﾔLPRINﾔI
:2946=4E 50 55 D4 4C 49 4E 50 /NPUﾔLINP
:294E=55 D4 49 C6 44 41 54 C1 /UﾔIﾆDATﾁ
:2956=52 45 41 C4 44 49 CD 52 /REAﾄDIﾍR
:295E=45 CD 45 4E C4 53 54 4F /EﾍENﾄSTO
:2966=D0 43 4F 4E D4 43 4C D3 /ﾐCONﾔCLﾓ
:296E=43 4C 45 41 D2 4F CE 4C /CLEAﾒOﾎL
```

この予約テーブルは,「通常のコマンド,ステートメント」,「拡張コマンド,ステートメント」,「関数」の3つの部分に分かれています。この区分にはFFHが置かれています(**表2-2**)。

**表2-3**に各命令中の中間コードを示します。"Store AD"は予約語が格納されている先頭アドレスです。拡張命令はFEH+1バイト,関数はFFH+1バイトの合計2バイトで構成されています。**リスト2-1**はこの中間コード表を表示させるプログラムです。ディスクの場合は予約語テーブルの先頭アドレスを2921Hに変えてください。

**表2-2 予約語テーブル・アドレス**

| | テープ | ディスク |
|---|---|---|
| 通常の命令 | 28F6～2AC9 | 2921～2AF4 |
| 拡張命令 | 2ACB～2BAA | 2AF6～2BD5 |
| 関数 | 2BAC～2CDD | 2BD7～2D0B |

**表2-3 中間コード**

| NO. | StoreAD | Word | Code |
|---|---|---|---|
| 1 | [28F6] | GOTO | -> 80 |
| 2 | [28FA] | GOSUB | -> 81 |
| 3 | [28FF] | GO | -> 82 |
| 4 | [2901] | RUN | -> 83 |
| 5 | [2904] | RETURN | -> 84 |
| 6 | [290A] | RESTORE | -> 85 |
| 7 | [2911] | RESUME | -> 86 |
| 8 | [2917] | LIST | -> 87 |
| 9 | [291B] | LLIST | -> 88 |
| 10 | [2920] | DELETE | -> 89 |
| 11 | [2926] | RENUM | -> 8A |
| 12 | [292B] | AUTO | -> 8B |
| 13 | [292F] | EDIT | -> 8C |
| 14 | [2933] | FOR | -> 8D |
| 15 | [2936] | NEXT | -> 8E |
| 16 | [293A] | PRINT | -> 8F |
| 17 | [293F] | LPRINT | -> 90 |
| 18 | [2945] | INPUT | -> 91 |
| 19 | [294A] | LINPUT | -> 92 |
| 20 | [2950] | IF | -> 93 |
| 21 | [2952] | DATA | -> 94 |
| 22 | [2956] | READ | -> 95 |
| 23 | [295A] | DIM | -> 96 |
| 24 | [295D] | REM | -> 97 |
| 25 | [2960] | END | -> 98 |
| 26 | [2963] | STOP | -> 99 |
| 27 | [2967] | CONT | -> 9A |
| 28 | [296B] | CLS | -> 9B |
| 29 | [296E] | CLEAR | -> 9C |
| 30 | [2973] | ON | -> 9D |
| 31 | [2975] | LET | -> 9E |
| 32 | [2978] | NEW | -> 9F |
| 33 | [297B] | POKE | -> A0 |
| 34 | [297F] | OFF | -> A1 |
| 35 | [2982] | WHILE | -> A2 |
| 36 | [2987] | WEND | -> A3 |
| 37 | [298B] | REPEAT | -> A4 |
| 38 | [2991] | UNTIL | -> A5 |
| 39 | [2996] | ??? | -> A6 |
| 40 | [2997] | ??? | -> A7 |
| 41 | [2998] | ??? | -> A8 |
| 42 | [2999] | TRON | -> A9 |
| 43 | [299D] | TROFF | -> AA |
| 44 | [29A2] | ??? | -> AB |
| 45 | [29A3] | ??? | -> AC |
| 46 | [29A4] | ??? | -> AD |
| 47 | [29A5] | DEFINT | -> AE |
| 48 | [29AB] | DEFSNG | -> AF |
| 49 | [29B1] | DEFDBL | -> B0 |
| 50 | [29B7] | DEFSTR | -> B1 |

```
 51  [29BD]  DEF        -> B2
 52  [29C0]  ???        -> B3
 53  [29C1]  LOAD       -> B4
 54  [29C5]  SAVE       -> B5
 55  [29C9]  MERGE      -> B6
 56  [29CE]  CHAIN      -> B7
 57  [29D3]  CONSOLE    -> B8
 58  [29DA]  WIDTH      -> B9
 59  [29DF]  OUT        -> BA
 60  [29E2]  SEARCH     -> BB
 61  [29E8]  WAIT       -> BC
 62  [29EC]  PAUSE      -> BD
 63  [29F1]  WRITE      -> BE
 64  [29F6]  SWAP       -> BF
 65  [29FA]  ERASE      -> C0
 66  [29FF]  ERROR      -> C1
 67  [2A04]  ELSE       -> C2
 68  [2A08]  CALL       -> C3
 69  [2A0C]  MON        -> C4
 70  [2A0F]  LOCATE     -> C5
 71  [2A15]  SCREEN     -> C6
 72  [2A1B]  KEY        -> C7
 73  [2A1E]  ???        -> C8
 74  [2A1F]  ???        -> C9
 75  [2A20]  LABEL      -> CA
 76  [2A25]  RANDOMIZE  -> CB
 77  [2A2E]  OPTION     -> CC
 78  [2A34]  LINE       -> CD
 79  [2A38]  OPEN       -> CE
 80  [2A3C]  CLOSE      -> CF
 81  [2A41]  SIZE       -> D0
 82  [2A45]  FIELD      -> D1
 83  [2A4A]  GET        -> D2
 84  [2A4D]  PUT        -> D3
 85  [2A50]  SET        -> D4
 86  [2A53]  FILES      -> D5
 87  [2A58]  LFILES     -> D6
 88  [2A5E]  DEVICE     -> D7
 89  [2A64]  NAME       -> D8
 90  [2A68]  KILL       -> D9
 91  [2A6C]  LSET       -> DA
 92  [2A70]  RSET       -> DB
 93  [2A74]  INIT       -> DC
 94  [2A78]  VDIM       -> DD
 95  [2A7C]  MAXFILES   -> DE
 96  [2A84]  ???        -> DF
 97  [2A85]  TO         -> E0
 98  [2A87]  STEP       -> E1
 99  [2A8B]  THEN       -> E2
100  [2A8F]  USING      -> E3
101  [2A94]  SUB        -> E4
102  [2A97]  BASE       -> E5
103  [2A9B]  TAB        -> E6
104  [2A9E]  SPC        -> E7
105  [2AA1]  EQV        -> E8
106  [2AA4]  IMP        -> E9
107  [2AA7]  XOR        -> EA
108  [2AAA]  OR         -> EB
109  [2AAC]  AND        -> EC
110  [2AAF]  NOT        -> ED
111  [2AB2]  ><         -> EE
112  [2AB4]  <>         -> EF
113  [2AB6]  =<         -> F0
114  [2AB8]  <=         -> F1
115  [2ABA]  =>         -> F2
116  [2ABC]  >=         -> F3
117  [2ABE]  =          -> F4
118  [2ABF]  >          -> F5
119  [2AC0]  <          -> F6
120  [2AC1]  +          -> F7
121  [2AC2]  -          -> F8
122  [2AC3]  MOD        -> F9
123  [2AC6]  ¥          -> FA
124  [2AC7]  /          -> FB
125  [2AC8]  *          -> FC
126  [2AC9]  ^          -> FD
127  [2ACA]             ->
128  [2ACB]  WINDOW     -> FE 80
129  [2AD1]  PSET       -> FE 81
130  [2AD5]  PRESET     -> FE 82
131  [2ADB]  COLOR      -> FE 83
132  [2AE0]  CIRCLE     -> FE 84
133  [2AE6]  POLY       -> FE 85
134  [2AEA]  PAINT      -> FE 86
135  [2AEF]  ???        -> FE 87
136  [2AF0]  POSITION   -> FE 88
137  [2AF8]  PATTERN    -> FE 89
138  [2AFF]  HCOPY      -> FE 8A
139  [2B04]  PLAY       -> FE 8B
140  [2B08]  SOUND      -> FE 8C
141  [2B0D]  BEEP       -> FE 8D
142  [2B11]  PRW        -> FE 8E
143  [2B14]  PALET      -> FE 8F
144  [2B19]  LAYER      -> FE 90
145  [2B1E]  CANVAS     -> FE 91
146  [2B24]  CREV       -> FE 92
147  [2B28]  CFLASH     -> FE 93
148  [2B2E]  CGEN       -> FE 94
149  [2B32]  CSIZE      -> FE 95
150  [2B37]  EJECT      -> FE 96
151  [2B3C]  CSTOP      -> FE 97
152  [2B41]  FAST       -> FE 98
153  [2B45]  REW        -> FE 99
154  [2B48]  APSS       -> FE 9A
155  [2B4C]  TVPW       -> FE 9B
156  [2B50]  CHANNEL    -> FE 9C
157  [2B57]  VOL        -> FE 9D
158  [2B5A]  CRT        -> FE 9E
159  [2B5D]  SCROLL     -> FE 9F
160  [2B63]  EFFECT     -> FE A0
161  [2B69]  GRAPH      -> FE A1
162  [2B6E]  MUSIC      -> FE A2
```

| No. | Address | Keyword | | Code |
|---|---|---|---|---|
| 163 | [2B73] | TEMPO | -> | FE A3 |
| 164 | [2B78] | CURSOR | -> | FE A4 |
| 165 | [2B7E] | VERIFY | -> | FE A5 |
| 166 | [2B84] | CLR | -> | FE A6 |
| 167 | [2B87] | LIMIT | -> | FE A7 |
| 168 | [2B8C] | KLIST | -> | FE A8 |
| 169 | [2B91] | ASK | -> | FE A9 |
| 170 | [2B94] | KBUF | -> | FE AA |
| 171 | [2B98] | CLICK | -> | FE AB |
| 172 | [2B9D] | BOOT | -> | FE AC |
| 173 | [2BA1] | DEVI$ | -> | FE AD |
| 174 | [2BA6] | DEVO$ | -> | FE AE |
| 175 | [2BAB] | | -> | |
| 176 | [2BAC] | INT | -> | FF 80 |
| 177 | [2BAF] | ABS | -> | FF 81 |
| 178 | [2BB2] | SIN | -> | FF 82 |
| 179 | [2BB5] | COS | -> | FF 83 |
| 180 | [2BB8] | TAN | -> | FF 84 |
| 181 | [2BBB] | LOG | -> | FF 85 |
| 182 | [2BBE] | EXP | -> | FF 86 |
| 183 | [2BC1] | SQR | -> | FF 87 |
| 184 | [2BC4] | RND | -> | FF 88 |
| 185 | [2BC7] | PEEK | -> | FF 89 |
| 186 | [2BCB] | ATN | -> | FF 8A |
| 187 | [2BCE] | SGN | -> | FF 8B |
| 188 | [2BD1] | FRAC | -> | FF 8C |
| 189 | [2BD5] | FIX | -> | FF 8D |
| 190 | [2BD8] | PAI | -> | FF 8E |
| 191 | [2BDB] | RAD | -> | FF 8F |
| 192 | [2BDE] | INP | -> | FF 90 |
| 193 | [2BE1] | CDBL | -> | FF 91 |
| 194 | [2BE5] | CSNG | -> | FF 92 |
| 195 | [2BE9] | CINT | -> | FF 93 |
| 196 | [2BED] | DSKF | -> | FF 94 |
| 197 | [2BF1] | EOF | -> | FF 95 |
| 198 | [2BF4] | FPOS | -> | FF 96 |
| 199 | [2BF8] | LOC | -> | FF 97 |
| 200 | [2BFB] | LOF | -> | FF 98 |
| 201 | [2BFE] | POS | -> | FF 99 |
| 202 | [2C01] | FAC | -> | FF 9A |
| 203 | [2C04] | SUM | -> | FF 9B |
| 204 | [2C07] | FRE | -> | FF 9C |
| 205 | [2C0A] | LPOS | -> | FF 9D |
| 206 | [2C0E] | STICK | -> | FF 9E |
| 207 | [2C13] | STRIG | -> | FF 9F |
| 208 | [2C18] | CHR$ | -> | FF A0 |
| 209 | [2C1C] | STR$ | -> | FF A1 |
| 210 | [2C20] | HEX$ | -> | FF A2 |
| 211 | [2C24] | OCT$ | -> | FF A3 |
| 212 | [2C28] | BIN$ | -> | FF A4 |
| 213 | [2C2C] | MKI$ | -> | FF A5 |
| 214 | [2C30] | MKS$ | -> | FF A6 |
| 215 | [2C34] | MKD$ | -> | FF A7 |
| 216 | [2C38] | SPACE$ | -> | FF A8 |
| 217 | [2C3E] | CGPAT$ | -> | FF A9 |
| 218 | [2C44] | KANJI$ | -> | FF AA |
| 219 | [2C4A] | ASC | -> | FF AB |
| 220 | [2C4D] | LEN | -> | FF AC |
| 221 | [2C50] | VAL | -> | FF AD |
| 222 | [2C53] | CVS | -> | FF AE |
| 223 | [2C56] | CVD | -> | FF AF |
| 224 | [2C59] | CVI | -> | FF B0 |
| 225 | [2C5C] | ??? | -> | FF B1 |
| 226 | [2C5D] | ??? | -> | FF B2 |
| 227 | [2C5E] | ERR | -> | FF B3 |
| 228 | [2C61] | ERL | -> | FF B4 |
| 229 | [2C64] | CSRLIN | -> | FF B5 |
| 230 | [2C6A] | STRPTR | -> | FF B6 |
| 231 | [2C70] | DTL | -> | FF B7 |
| 232 | [2C73] | ??? | -> | FF B8 |
| 233 | [2C74] | ??? | -> | FF B9 |
| 234 | [2C75] | LEFT$ | -> | FF BA |
| 235 | [2C7A] | RIGHT$ | -> | FF BB |
| 236 | [2C80] | MID$ | -> | FF BC |
| 237 | [2C84] | INKEY$ | -> | FF BD |
| 238 | [2C8A] | INSTR | -> | FF BE |
| 239 | [2C8F] | HEXCHR$ | -> | FF BF |
| 240 | [2C96] | MEM$ | -> | FF C0 |
| 241 | [2C9A] | SCRN$ | -> | FF C1 |
| 242 | [2C9F] | VARPTR | -> | FF C2 |
| 243 | [2CA5] | STRING$ | -> | FF C3 |
| 244 | [2CAC] | TIME | -> | FF C4 |
| 245 | [2CB0] | DAY$ | -> | FF C5 |
| 246 | [2CB4] | DATE$ | -> | FF C6 |
| 247 | [2CB9] | FN | -> | FF C7 |
| 248 | [2CBB] | USR | -> | FF C8 |
| 249 | [2CBE] | ??? | -> | FF C9 |
| 250 | [2CBF] | ??? | -> | FF CA |
| 251 | [2CC0] | ATTR$ | -> | FF CB |
| 252 | [2CC5] | POINT | -> | FF CC |
| 253 | [2CCA] | CHARACTER$ | -> | FF CD |
| 254 | [2CD4] | CMT | -> | FF CE |
| 255 | [2CD7] | MIRROR$ | -> | FF CF |

リスト2-1

```
100 '
110 ' ﾁｭｳｶﾝ ｺｰﾄﾞ      X1 HuBASIC (TAPE)
120 '
130 CLS:WIDTH 40
140 AD=&H28F6    'Word Table Top Address
150 LF=0         'PRINT OUT FLAG
160 LL=50        'PRINT LINE
170 DIM F$(2)
180 F$(0)="":F$(1)="FE ":F$(2)="FF "
190 SP$=SPACE$(10)
200 DEF FNH$(X)=RIGHT$("000"+HEX$(X),4)
210 MSG$="NO. StoreAD  Word          Code"
220 IF LF<>0 THEN LPRINT MSG$:LPRINT
230 PRINT MSG$
240 '
250 CD=&H80:F=0:CT=1:LCT=1
260 REPEAT
270   GOSUB "MAIN"
280   CT=CT+1:CD=CD+1:LCT=LCT+1
290 UNTIL CT=256
300 END
310 '
320 LABEL        "MAIN"
330 WD$="":STAD=AD
340   D=PEEK(AD):AD=AD+1
350   IF D=&H80 THEN WD$="???":GOTO 390
360   IF D=&HFF THEN WD$="     ":GOTO 390
370   XD=(D AND &H7F)
380   WD$=WD$+CHR$(XD)
390 IF D<&H80 THEN 340
400 '
410 IF LF<>0 THEN LPRINT USING "###  ";CT;
420 PRINT USING "###  ";CT;
430 P$="["+FNH$(STAD)+"]  "
440 P$=P$+LEFT$(WD$+SP$,10)+"-> "
450 IF D=&HFF THEN F=F+1:CD=&H7F:LCT=0:GOTO 470
460 P$=P$+F$(F)+RIGHT$("0"+HEX$(CD),2)
470 PRINT P$
480 IF LF=0 THEN 540
490   LPRINT P$
500   IF D=&HFF THEN LPRINT CHR$(12):GOTO 530
510   IF (LCT MOD LL)>0 THEN 540
520   FOR I=1 TO 64-LL:LPRINT:NEXT I
530   LPRINT MSG$:LPRINT
540 RETURN
```

# 2-4 定数の格納形式と内部コード

BASICテキストの中には中間コードとは別にリストに表示されない内部コードがあります。内部コードは主に数値定数を格納するために使われます。内部コードの一覧を**表2-4**に示します。内部コードには次のようなものがあります。

**表2-4 内部コード**

| 内部コード | 意　味 |
|---|---|
| 00H | 行エンド，各エリアのエンド |
| 01H～0AH | 定数　0～9<br>ただし，01H＝0，02H＝1，…，0AH＝9 |
| 0BH ×× ×× | 行番号値　　(16進2バイト) |
| 0CH ×× ×× | 行番号実アドレス　　(16進2バイト) |
| 0DH ×× ×× | 8進数　&O　　(16進2バイト) |
| 0EH ×× ×× | 2進数　&B　　(16進2バイト) |
| 0FH ×× ×× | 16進数　&H　　(16進2バイト) |
| 12H ×× ×× | 10～23767の整数　　(16進2バイト) |
| 15H ×× ××<br>×× ×× ×× | 単精度実数　　(浮動小数点表現) |
| 18H ×× ××<br>×× ×× ××<br>×× ×× ×× | 倍精度実数　　(浮動小数点表現) |

●行エンド・コード（00H）

Aレジスタの値が00Hかどうかを調べるには"OR A"または，"AND A"で簡単にかつ高速に調べられます。そのため，頻繁に出てくる行の終わりのコードには00Hが使われています。

●定数0～9（01H～0AH）

00Hがエンド・コードに使われているので，1つずつずれて，01Hが0を0AHが9を表します(**図2-7**)。

●行番号値 (0BH, ＊＊, ＊＊)

0BH に続く2バイトが行番号を16進に変更した値を示します。プログラムを入力した時点での格納形式であり，1度でも実行されると,次の行番号実アドレスに変換されます(図2-8a)。

●行番号実アドレス (0CH, ＊＊, ＊＊)

0CH に続く2バイトは指定された行番号が入っている実際のアドレスであることを意味します(図2-8b)。このように変更しておけば，その度に行番号を探す手間がはぶけます。

図2-7 定数0～9の内部表現例

```
10 A=1:A=2:A=3:A=9

:9FC5=14 00 0A 00 41 F4 02 3A /....A日.:
:9FCD=41 F4 03 3A 41 F4 04 3A /A日.:A日.:
:9FD5=41 F4 0A 00 00 00 00 01 /A日......
```

図2-8 行番号値と行番号実アドレス

```
10 GOSUB 20
20 RETURN 10
30 GOTO 30

(a)実行前

:9FC5=0A 00 0A 00 81 20 0B 14 /....▬...
:9FCD=00 00 0A 00 14 00 84 20 /......■
:9FD5=0B 0A 00 00 0A 00 1E 00 /........
:9FDD=80 20 0B 1E 00 00 00 00 /_ ......

(b)実行後

:9FC5=0A 00 0A 00 81 20 0C CF /....▬ .マ
:9FCD=9F 00 0A 00 14 00 84 20 /ﾉ.....■
:9FD5=0C C5 9F 00 0A 00 1E 00 /.ﾅﾉ.....
:9FDD=80 20 0B 1E 00 00 00 00 /_ ......
           ↖
            この行は実行されていない
```

ですから，入力した直後のプログラムより，１度実行したプログラムの方がスピードが速くなります。

● 8進，2進，16進数（0DH～0FH）

続く2バイトが8進，2進，16進の各数体系の定数であることを意味します。0DH＝"&O"，0EH＝"&B"，0FH＝"&H" の中間コードであることを考えればわかりやすいでしょう（**図2-9**）。

**図2-9　2進，8進，16進の内部表現例**

```
10 A=&O567
20 A=&B101110111
30 A=&H177


:9FC5=0A 00 0A 00 41 F4 0D 77 /....A日.w
:9FCD=01 00 0A 00 14 00 41 F4 /......A日
:9FD5=0E 77 01 00 0A 00 1E 00 /.w......
:9FDD=41 F4 0F 77 01 00 00 00 /A日.w....
```

いずれも10進に変換する場合は２の補数表現として扱われるので，格納できる値の範囲は－32768～32767の整数値だけです。

●10進定数（12H，＊＊，＊＊または15H，＊＊，＊＊，＊＊，＊＊，＊＊または18H，＊＊，＊＊，＊＊，＊＊，＊＊，＊＊，＊＊，＊＊）

12H は続く２バイトが10～32768の10進で表示されることを示します。負の数は 12H の前に F8H をつけて区別しています（**図2-10**）。

15H は続く５バイトが，18H は続く８バイトが浮動小数点表記の実数であることを示しています（**図2-11**）。

図2-10 負数の内部表現例

```
10 A=10
20 A=-10

:9FC5=0A 00 0A 00 41 F4 12 0A /....A日..
:9FCD=00 00 0B 00 14 00 41 F4 /......A日
:9FD5=F8 12 0A 00 00 00 00 00 / .......
```

図2-11 浮動小数点の内部表現例

```
10 A=14.5:A=.1
20 A=123456789#

:9FC5=16 00 0A 00 41 F4 15 84 /....A日.■
:9FCD=68 00 00 00 3A 41 F4 15 /h...:A日.
:9FD5=7D 4C CC CC CD 00 10 00 /}Lフフヘ...
:9FDD=14 00 41 F4 18 9B 6B 79 /..A日.┘ky
:9FE5=A2 A0 00 00 00 00 00 00 /「 ......
```

# 2-5 浮動小数点

普段扱う数値の中には，有効数字は少なくても桁数の多いものがあります。たとえば，1200000円と言ったお金などその良い例です。コンピュータでこれらをそのままの形でメモリに格納したのでは，ゼロの分だけメモリの無駄使いとなります。そこで，これを有効数字と位どりの部分に分けて，$1.2\times10^6$ などと表現し，これを浮動小数点表現と呼んでいます。これは仮数部$\times10^{指数部}$となっており，コンピュータでも少ないメモリで済むという利点があります。

人間が扱う数は10進数が普通ですが，コンピュータが扱いやすい数は２進数です。ですから，このような表現も，仮数部$\times 2^{指数部}$とするのが自然です。２進数の浮動小数点表現には，いくつかの方法がありますが，HuBASICの表現例として，10進の14.5を変換してみましょう。これは，

$$
\begin{aligned}
14.5 &= 8+4+2+0.5 \\
&= 2^3+2^2+2^1+2^{-1} \\
&= 1110.1\mathrm{B} \\
&= 1.1101\mathrm{B}\times 2^3
\end{aligned}
$$

となります。０以外の実数はすべてこの形式に変換することができ，一般的に次のように書けます。

$\pm 1.{*}{*}{*}{*}{*}{*}{*}\mathrm{B}\times 2^{\pm n}$　　＊は０か１

指数部に１バイト割り当て，０のときは指数部を00Hと決めています。指数部は正負両方を表さなければならないので，81Hを加え，81H以上を正の指数，以下を負の指数とします。つまり，82Hは１，83Hは２，…FFHは126で，80Hは－１，7FHは－２，…01Hは－128となります。

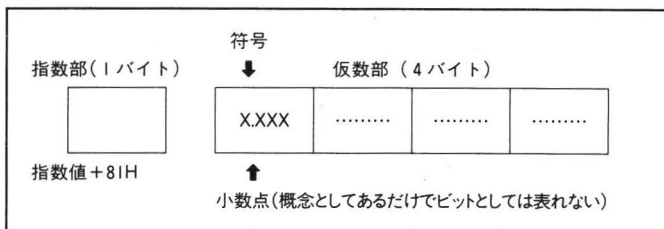
次に仮数部の表現方法ですが，仮数部の整数部は常に１なるように変換したので，このビットは格納する必要はありま

せん。このあまったビットは仮数部の符号として利用しています。つまり，仮数部の最上位ビットが0のとき正，1のとき負になります。

単精度数値を格納するために，X1では仮数部に4バイト使っているために全体では**図2-12**のようになります。よって例としてあげた値14.5（$=1.1101B\times 2^3$）は，

| | |
|---|---|
| 指数部 | 3＋81H＝84H |
| 仮数部 | 01101000 00000000 00000000 00000000 |
| | ＝68H 00H 00H 00H |

と表現されます（**図2-11**）。

**図2-12 単精度浮動小数点**

さて，次に10進で0.1という数値を考えてみます。これは，$0.1 = 1.1001100110011100\cdots\times 2^{-4}$と無限循環小数となってしまいます。このような場合は有効ビットを0捨1入します。よって，0.1は

| | |
|---|---|
| 指数部 | −4 ＋81H＝7DH |
| 仮数部 | 01001100 11001100…11001100 1100…（桁上げ） |
| | ＝4CH CCH CCH CDH |

と表されます。

浮動小数点表現による格納形式を調べる簡単なプログラムを**リスト2-2**に示します。

倍精度数値の場合は，仮数部が7バイトに増えるだけで，まったく同じ形式です。

リスト2 2

```
10 ' ﾌﾄﾞｳ ｼｮｳｽｳﾃﾝ ( ﾅｲﾌﾞ ﾋｮｳｹﾞﾝ )
20 A=.1
30 X=VARPTR(A)
40 PRINT A;" = ";
50 FOR I=0 TO 4
60   D=PEEK(X+I)
70   PRINT RIGHT$("0"+HEX$(D),2);" ";
80 NEXT I
```

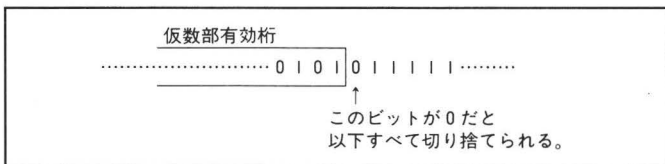
**実行結果**

```
.1  = 7D 4C CC CC CD
```

## 2-5-1 浮動小数点演算誤差

浮動小数点演算の場合，2進数と10進数の表記の違いからどうしても変換時の誤差が生じます。先の例では0.1が無限循環小数になってしまい，途中で切り上げなければなりませんでした。

通常の浮動小数点演算では，切り捨てられる確率の方が高いため，繰り返し加算されていくと実際の値より小さくなってしまうことが多いようです（**図2-13**）。

図2 13 **切り捨て**

マニュアルでも演算誤差について簡単に述べられています。HuBASIC では単精度数値の仮数部に4バイトも使っています(他の BASIC では仮数部3バイト，表示数6桁が多い)。そのため，10進有効桁は9桁まで表示できるのですが，外部表現では8桁しか表示しません。このような方法で外部に誤差がでないように工夫されています。

それでも，0.1を繰り返し加えていくと459回目で誤差が外

部表現に表れてしまいます。この対策として，マニュアルでは「10倍して整数化した後に演算を行い，10で割って正しい値を得る」方法が紹介されましたが，文字型変換による誤差補正は説明不足のようです。**リスト2-3**がその例で50行で誤差補正を行っています。この方法がもっとも直接的な表現でプログラムが書けて，誤差も出ないようになりますが，実行速度がかなり落ちるのが難点です。

リスト2-3

```
10 ' ｴﾝｻﾞﾝ ｺﾞｻ ﾎｾｲ
20 A=0
30 FOR I=0 TO 1500
40   A=A+.1
50   A=VAL(STR$(A))
60   PRINT I+1,A
70 NEXT
```

# 2-5-2 システム最小値

統計計算や数学の微積分・極限を扱う場合，ゼロではないが，ゼロに非常に近い値が問題になります。BASICの扱える数にも限度があるので，その値が決まっています。この値をそれぞれのBASICの機械最小値とかシステム最小値と呼んでいます。システム最小値以下の値はゼロと扱われるので，こういった数値を扱う方は十分注意してください。

**リスト2-4**は単精度でのシステム最小値を求めるプログラムで，**図2-14**は実行結果です。実行後ゼロになる寸前の値，

$2.9387359 \times 10^{-39}$

がHuBASICでのシステム最小値です。これを内部表現に直すと，

| 指数部 | 仮数部 |
|---|---|
| 01H | 00H 00H 00H 00H |

となります。仮数部がすべてゼロなのは少し不思議な感じもします。実行途中で表示された0.5や0.25は2進変換を行って

も変換誤差の生じない値です。このような値だけの場合は誤差補正は不用です。

リスト2-4

```
10   システム サイショウチ
20 DEFSNG A
30 A=1
40 A=A/2 : PRINT A
50 IF A<>0 THEN 40
```

図2-14 リスト2-4の実行結果

```
.5
.25
.125
.0625
.03125
.015625
.0078125
.00390625
1.953125E-03
9.765625E-04
4.8828125E-04
      :
      :
1.5046328E-36
7.5231639E-37
3.7615819E-37
1.880791E-37
9.4039548E-38
4.7019774E-38
2.3509887E-38
1.1754944E-38
5.8774718E-39
2.9387359E-39
0
```

HuBASIC のエディタではシステム最小値を直接打ち込むことはできないようです。たとえば,

```
10 A=2.9387359E-39
```

とするとオーバーフローエラーとなってしまいます。変数に代入するには次のようにするとよいでしょう。

```
10 A=2^-126/4
```

ちなみに倍数精度数値のシステム最小値は,

$2.938735877055719 \times 10^{-39}$

です。

## 2-5-3 浮動小数点演算サブルーチン

HuBASIC には**表2-5**のような浮動小数点演算用のサブルーチンが用意されています。USR 命令などで単精度・倍精度実数演算が必要なときに役立ててください。また,浮動小数点演算のプログラム参考例として解析するにも良い材料となるでしょう(ただし,かなり複雑です)。表のカッコ内の数字はディスク BASIC のアドレスです。

各サブルーチンをコールする前にデータの種類(単精度,倍精度の区別)を入れておくアドレスが決まっていて,テープ・バージョンの場合 9CF8H,ディスク・バージョンの場合,A5DBH です。ここの値は次のような意味を持っています。

```
02H・・・整数(16進2バイト)
05H・・・単精度実数(浮動小数点表現5バイト)
08H・・・倍精度実数(浮動小数点表現 8バイト)
03H・・・文字列
```

浮動小数点演算用サブルーチンなのに整数や文字列の値が入るのはおかしいと思った方もいるでしょうが,実はこれらのサブルーチンは整数演算・文字列演算をも兼ねている場合が

多いのです。この9CF8H (A5DBH) の内容をみて各処理へ分かれていきます。

**表2-6**は，浮動小数点で格納されている定数の先頭アドレスで単精度，倍精度とも共通に利用できます。

**表2-5 浮動小数点演算サブルーチン**

( )内はディスク・バージョン

| 機　能 | アドレス | 内　容 |
|---|---|---|
| 加算 | 9161 (9A44) | ●HL と DE で示される浮動小数点データを加えて HL の示すアドレスへ格納する。整数，文字列の加算も兼ねている。オーバーフロー時はエラー処理へジャンプ。<br><br>●入力　HL …被加数データ先頭アドレス<br>DE …加数データ先頭アドレス<br>(9CF8)…データの種類<br><br>●出力　(HL)…加算結果<br><br>●レジスタ… HL , DE, IY は保存 |
| 減算 | 9158 (9A3B) | ●HL の示す値から DE の示す値を引いて HL の示すアドレスへ格納。オーバーフロー時はエラー処理へジャンプ。<br><br>●入力　HL…被減数データ先頭アドレス<br>DE…減数データ先頭アドレス<br>(9CF8)…データの種類<br><br>●出力　(HL)…減算結果<br><br>●レジスタ… HL, DE, IY は保存 |
| 乗算 | 9712 (9FF5) | ●HLとDEで示されるデータをかけHLが示すアドレスに格納する。整数の乗算もかねている。オーバーフロー時はエラー処理へジャンプ。<br>●入力　HL…被乗数データ先頭アドレス<br>DE…乗数データ先頭アドレス<br>(9CF8)…データの種類<br><br>●出力　(HL)…乗算結果<br>●レジスタ　HL, DE, IYは保存 |

| 機　能 | アドレス | 内　　容 |
| --- | --- | --- |
| 除算 | 9807 (A0EA) | ●HL の示す値を DE の示す値で割り，HL の示すアドレスへ格納。整数の演算もかねている。オーバーフロー時または (DE) がゼロのときはエラー処理へジャンプ。<br><br>●入力　HL …被除数データ先頭アドレス<br>DE …除数データ先頭アドレス<br>(9CF8)…データの種類<br><br>●入力　(HL)…除算結果<br><br>●レジスタ… HL, DE は保存 |
| 剰余 | 905F(9C39) | ●HL の示す値で割ったときの余を求め，HL の示すアドレスに格納。整数の剰余計算も兼ねている。オーバーフロー時はエラー処理へジャンプ。<br><br>●入力　HL….被剰数データ先頭アドレス<br>DE …剰数データ先頭アドレス<br>(9CF8)…データの種類<br><br>●出力　(HL)…剰余結果<br><br>●レジスタ　HL, DE は保存 |
| 比較 | 9356(9C39) | ●HL とDE で示される浮動小数点データを比較し，フラグを変化させる。整数，文字列の比較も兼ねている。<br><br>●入力　HL …比較データの先頭アドレス<br>DE …比較データの先頭アドレス<br>(9CF8)…データの種類<br><br>●出力　(HL)＝(DE)…ゼロフラグ・セット<br>(HL)＜(DE)…キャリーフラグ・セット<br>(HL)＞(DE)…キャリーフラグ・リセット<br><br>●レジスタ　HL, DE, IX, IY, 裏レジスタは保存 |

| 機　能 | アドレス | 内　容 |
|---|---|---|
| 1を加算 | 8A27(930A) | ●HL で示される値に1加える。オーバーフロー時はエラー処理へジャンプ。<br>●入力　HL …データ先頭アドレス<br>(9CF8)…データの種類<br>●出力　(HL)…加算結果<br>●レジスタ　HL, IY は保存 |
| 1を減算 | 8A21(9304) | ●HL で示される値から1引く。オーバーフロー時はエラー処理へジャンプ。<br>●入力　HL …データの先頭アドレス<br>(9CF8)…データの種類<br>●出力　(HL)…減算結果<br>●レジスタ　HL, IY は保存 |
| 1と比較 | 8A2D(931D) | ●HL で示される値と1とを比較し，フラグを変化させる。<br>●入力　HL …データ先頭アドレス<br>(9CF8)…データの種類<br>●出力　(HL)＝1…ゼロフラグ・セット<br>(HL)<1…キャリーフラグ・セット<br>(HL)>1…キャリーフラグ・リセット<br>●レジスタ　HL, IX, IY および裏レジスタ保存 |
| 2倍 | 890A(91D7) | ●HLで示される値を2倍する。オーバーフロー時はエラー処理へジャンプ。<br>●入力　HL …データ先頭アドレス<br>●出力　(HL)…2倍されたデータ<br>●レジスタ　全レジスタ保存 |

| 機　能 | アドレス | 内　容 |
| --- | --- | --- |
| 1/2倍 | 890F(91DC) | ●HL で示される値を2で割る。オーバーフロー時はエラー処理へジャンプ。<br>●入力　HL …データ先頭アドレス<br>●出力　(HL)…1/2倍されたデータ<br>●レジスタ　A 以外保存 |

**表2-6 各種定数エントリー**

( )内はディスク・バージョン

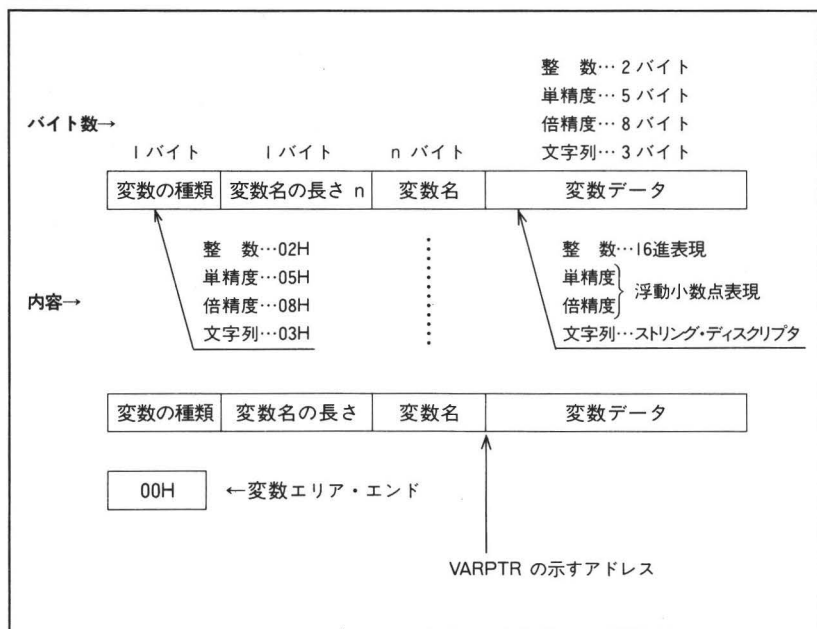
| アドレス | 内　容 |
| --- | --- |
| 5BA5(5BE5) | 1000000000000000D＝千兆 |
| 5BAD(5BED) | 100000000000000D＝百兆 |
| 5BB5(5BF5) | 10000000000000D＝十兆 |
| 5BBD(5BFD) | 1000000000000D＝一兆 |
| 5BC5(5C05) | 100000000000D＝千億 |
| 5BCD(5C0D) | 10000000000D＝百億 |
| 5BD5(5C15) | 1000000000D＝十億 |
| 5BDD(5C1D) | 100000000D＝一億 |
| 5BE5(5C25) | 10000000D＝一千万 |
| 5BED(5C2D) | 1000000D＝百万 |
| 5BF5(5C35) | 100000D＝十万 |
| 5BFD(5C3D) | 10000D＝一万 |
| 5C05(5C45) | 1000D＝千 |
| 5C0D(5C4D) | 100D＝百 |
| 5C15(5C55) | 10D＝十 |
| 5C1D(5C5D) | 0 |
| 5C25(5C65) | 0.1 |
| 8E14(96F7) | π　単精度　3.1415927<br>倍精度　3.141592653589793 |
| 8DCF(96B2) | π/4 |
| 9073(9956) | 2/5 |
| 907B(995E) | 5/3 |
| 9103(99E6) | 1/120 |

# 2-6 変数の格納形式

テキストとは別に変数の値を格納しておくエリアが，テキストのすぐ後ろに用意されています。このエリアはプログラムをキー入力した時点ではまだ存在せず，実行すると出てきた変数の順に格納されます。

## 2-6-1 単純変数

変数の種類，変数名の長さによって必要なバイトは異なりますが，各変数とも同じ形式で格納されます(**図2-15**)。1バイト目の変数の種類の値はそのまま変数データの格納バイト数も表しています。

図2-15 **変数の格納形式**

変数データのところは，変数が整数型のときは16進2バイトでそのまま入っており，単精度，倍精度型のときは，先に説明した浮動小数点表現で格納されています。

BASICのVARPTR関数で与えられる値は，指定された各変数の値が入っている先頭アドレスです。ですから**図2-16**の場合，

```
PRINT HEX$ (VARPTR (A %))
```

を実行すると9FF2Hという値になります。9FF2Hには変数A％の値1（0001H）が入っています。

**図2 16 変数の格納形式例**

```
10 A%=1
20 A!=2
30 A#=3
40 A$="ABC"

:9FC5=09 00 0A 00 41 25 F4 02 /....A%日.
:9FCD=00 09 00 14 00 41 21 F4 /.....A!日
:9FD5=03 00 09 00 1E 00 41 23 /......A#
:9FDD=F4 04 00 0D 00 28 00 41 /日....(.A
:9FE5=24 F4 22 41 42 43 22 00 /$日"ABC".
:9FED=00 00 02 01 41 01 00 05 /....A...
:9FF5=01 41 82 00 00 00 00 08 /.A▬.....
:9FFD=01 41 82 40 00 00 00 00 /.A▬@....
:A005=00 00 03 01 41 03 22 02 /....A.".
:A00D=00 00 4F 39 5D 00 00 00 /..09]...
```

変数データの中で，文字変数は他と異なり，**図2-17**に示すようなストリング・ディスクリプタが格納されています。実際の文字列データはストリング・データ・バッファに格納されています。ストリング・ディスクリプタの2バイト目，3バイト目にはファイル用ストリング・バッファの先頭番地を0000Hとした相対アドレスが入っています。

HuBASICにはSTRPTRというシステム定数があり，これがファイル用ストリング・バッファの先頭アドレスを持っています。よってBASICでの計算式はAD=VARPTR

(A$) とした場合,

```
HEX$(STRPTR+PEEK(AD+1)+PEEK(AD+2)*256)
```

で求められます。今回の例では A230H というアドレスが得られたので,これをダンプしてみると,

```
:A230=41 42 43 00 00 41 32 33/ABC..A23
```

となっていました。後ろの方の「A23」の文字はアドレス計算を行ったときに使われたものです。

**図2 17 文字列,数値の格納形式**

(a) 文字列

ストリング・ディスクリプタ

| 1バイト | 2バイト |
|---|---|
| 文字数 | STRPTR を 0000H としたときの相対アドレス |

(b) 数値

| | 2バイト | |
|---|---|---|
| 整　数 | 下位 | 上位 |

| | 1バイト | 4バイト |
|---|---|---|
| 単精度 | 指数部 | 仮 数 部 |

| | 1バイト | 7バイト |
|---|---|---|
| 倍精度 | 指数部 | 仮 数 部 |

# 2-6-2 配列変数

配列変数の格納形式は少々複雑で，**図2-18**のような構造になっています。

n次元配列の場合は，「配列の大きさ」の部分がn個に増えます。そして，その格納順序は後に出てくる引数ほど先に格納されます。変数データの格納順序は逆で，先にある引数が先にカウントされます。実例で見てみましょう。**図2-19**はプログラムと実行後のダンプ・リストです。A（3，4）と宣言した場合は引数4（配列の大きさは5）が先に格納され，次に引数3（大きさは4）が格納されます。

変数データは，A(0，0)のデータは当然いちばん先頭ですが，次にA（1，0）のデータが格納されます（**表2-7**）。

変数の種類は単純変数と区別するために，最上位ビットを立てています。

**図2-18 配列の格納形式**

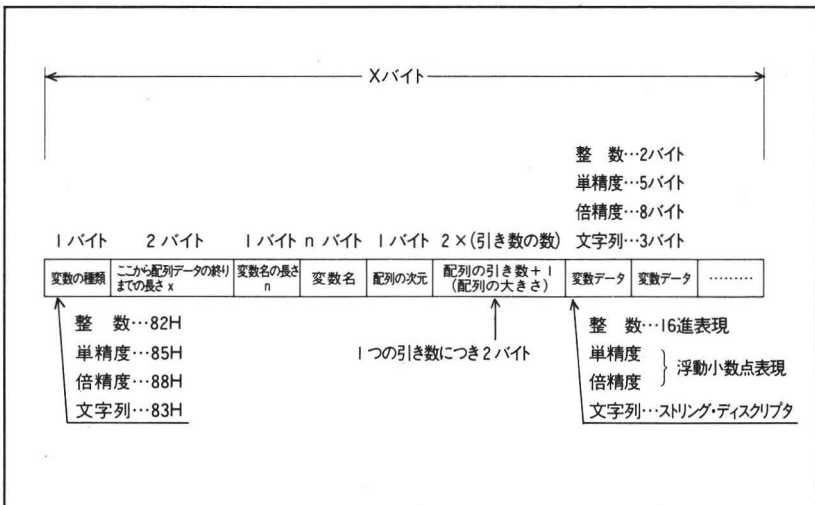



図2-19 配列の格納形式例

```
10 DEFINT A
20 DIM A(3,4)
30 A(0,0)=1
40 A(1,0)=2
50 A(1,1)=3


:9FC5=08 00 0A 00 AE 20 41 00 /....ョ A.
:9FCD=0D 00 14 00 96 20 41 28 /....| A(
:9FD5=04 2C 05 29 00 0D 00 1E /.,.)....
:9FDD=00 41 28 01 2C 01 29 F4 /.A(.,.)日
:9FE5=02 00 0D 00 28 00 41 28 /....(.A(
:9FED=02 2C 01 29 F4 03 00 0D /.,.)日...
:9FF5=00 32 00 41 28 02 2C 02 /.2.A(.,.
:9FFD=29 F4 04 00 00 00 82 31 /)日....▬1
:A005=00 01 41 02 05 00 04 00 /..A.....
:A00D=01 00 02 00 00 00 00 00 /........
:A015=00 00 03 00 00 00 00 00 /........
:A01D=00 00 00 00 00 00 00 00 /........
:A025=00 00 00 00 00 00 00 00 /........
:A02D=00 00 00 00 00 00 00 00 /........
```

表2-7 配列の格納状態

| アドレス | データ | 意　味 |
|---|---|---|
| A003 | 82 | 変数の種類＝整数型配列変数 |
| A004 | 31 00 | 配列データの終りまでの長さ＝0031バイト |
| A006 | 01 | 変数名の長さ　＝1バイト |
| A007 | 41 | 変数名　＝「A」 |
| A008 | 02 | 配列の次元　＝2次元 |
| A009 | 05 00 | 2つ目の引き数＋1　＝5 |
| A00B | 04 00 | 1つ目の引き数＋1　＝4 |
| A00D | 省略 | A (0,0)　A (1,0)　A (2,0)　A (3,0)　のデータ |
| A015 | 〃 | A (0,1)　A (1,1)　A (2,1)　A (3,1)　のデータ |
| A01D | 〃 | A (0,2)　A (1,2)　A (2,2)　A (3,2)　のデータ |
| A025 | 〃 | A (0,3)　A (1,3)　A (2,3)　A (3,3)　のデータ |
| A02D | 〃 | A (0,4)　A (1,4)　A (2,4)　A (3,4)　のデータ |

# 2-7 マシン語プログラムとのリンク

マシン語プログラムを呼び出すためにHuBASICには CALL命令とUSR命令が用意されています。

## 2-7-1 CALL命令

CALL命令はその直後に書かれたアドレスにあるマシン語ルーチンをコールします。マシン語のRET命令(C9H)を実行することでBASICプログラムに戻ってきて動作を継続します。

通常のBASICでのCALL命令にはマシン語ルーチンとのデータの受け渡し機能は付いていません。ですからPEEK, POKE命令でメモリを介してデータの受け渡しを行うことになります。

しかし, HuBASICではCALL命令に入力パラメータの機能を付けることができます(マニュアルには載っていない)。これには, 次のようにアドレスの後ろにカッコで囲んで数値を書きます。この値はHLレジスタに渡されます。

| 形式：CALL　アドレス（入力パラメータ） |
|---|
| 例：CALL　&HC000（&H0030） |

ただし, マシン語ルーチンからBASICへデータを返す機能はないので, この場合はPEEK命令で行います。また, マシン語ルーチン内でレジスタを保存する必要はありません。

## 2-7-2 USR命令

USR 命令はCALL 命令に比べて少々理解しづらいようですが，パラメータの受け渡しが比較的容易にできる点で優れています。

USR 命令は10個のアドレスを割りあてることができ，あらかじめ DEF USRn＝コール・アドレスで定義しておきます(n は 0 ～ 9，n＝ 0 のときは省略可)。

このアドレスは 1 度定義されると NEW や CLEAR を実行しても消えません。

USR のコール・アドレス・テーブルは 9D32H～9D49H にあります（ディスク・バージョンは A615H～A628H)。

初期値は 9C9FH（ディスク：A582H）で，これは『Undefined function』のエラーのエントリーです。

USR 命令は関数ですから，

```
形式：変数名＝USRn（数値または数値変数）
      変数名＝USRn（文字列または文字変数）
例：  A＝USR（P）
      A$＝USR2（"A"）
```

といった形式になります。

入力パラメータと，出力パラメータに入れる変数は，変数の種類が数値または文字に統一されていなければなりません。A＝USR(P)などを実行すると，あらかじめ USRn に定義されていたアドレスがコールされ，そこのマシン語ルーチンが実行されます。

カッコ内におかれた変数または定数が入力パラメータで，マシン語ルーチンに数値や文字を渡すためのものです。

マシン語ルーチンから BASIC に数値や文字を返すための出力パラメータは，A または A$ (変数名はなんでもよい) に入って BASIC に渡されます。

マシン語プログラムに送られるレジスタの意味は，入力パラメータの種類によって異なります。**表2-8**に各種パラメータに対するレジスタの内容をまとめておきました。

**表2-8 USR関数で渡されるレジスタの意味**

| パラメータ／レジスタ名 | 数値型 | 文字型 |
|---|---|---|
| A | データの種類・バイト数<br>02H …整数 05H …単精度 08H …倍精度 | データの種類・バイト数<br>○03H …文字型のみ |
| HL | FAC 先頭アドレス<br>○FAC には数値が格納されている | FAC 先頭アドレス<br>○FAC にはストリング･ディスクリプタが格納されている |
| B | 無意味 | 文字列の長さ<br>○　不定 |
| DE | 無意味 | 文字列データ格納先アドレス<br>○　不定 |
| IX | エラー処理ルーチン・エントリーアドレス<br>テープ・バージョン　：2064H<br>ディスク・バージョン：2076H | |

マニュアルの『USR命令の使い方』のところでわかりづらい点が1カ所あります。それは，USR命令のパラメータに文字変数のみを使用した場合，パラメータの文字変数の内容までが変わってしまう，というところです。しかし，USR関数のマシン語プログラムの中でレジスタをどんなに変えてもパラメータ文字変数の内容は破壊されません。

ここでの説明は，DEレジスタで示されるメモリの内容(パラメータ変数の文字データそのもの）を書き換えたときに起こることです。ですから，パラメータの文字列の内容を読み出すだけのときは気にしなくて良いのです。

もしも，書き換える必要があって，なおかつパラメータ文字変数を変えたくないときは，

```
A$=USR (P$+"")
```

のように文字定数（ヌルコードで良い）との演算を行わせます。こうすることによってストリング・データ・バッファ内にもう1つ，パラメータ文字変数と同じ内容の文字列がコピ

ーされ，そちらの先頭アドレスがDEレジスタに入って渡されることになります。コピーされた所を書き換えたとしても元の文字列は保存されるわけです。**リスト2-5**にその例を示します。これはP$をパラメータとし，その1文字目をスペースに変えるプログラムです。A$=USR(P$)としたのでは，パラメータであるP$の内容までもが変わってしまいますが，A=USR(P$+"")とした方はP$は"ABC"のまま変化しません。

USR命令で受け渡される真の変数データはすべてメモリ上にあります。とくにHLレジスタで示されるエリア(HLにはFACの先頭番地が入る）を介して受け渡しを行うのが一般的です。

コールされた時点ではFACエリア上にパラメータのデータが格納されています。この格納形式は**2-6**で説明した変数エリアの格納形式と同じです（**図2-17**)。

マシン語プログラムからリターンした後はFACエリアに

**リスト2-5**

```
100 ' USR ﾊﾟﾗﾒｰﾀ
110 CLEAR &HBFFF:C$=CHR$(34)
120 AD=&HC000:DEF USR=AD
130 FOR I=0 TO 999
140   READ D$:IF D$="END" THEN 180
150   POKE AD+I,VAL("&H"+D$)
160 NEXT I
170 ' -- 1ﾓｼﾞﾒ ｦ ｽﾍﾟｰｽ ﾆ ｶｷｶｴﾙ
180 DATA 3E,20  :'LD    A,20H
190 DATA 12     :'LD    (DE),A
200 DATA C9     :'RET
210 DATA END
220 CLS:LPRINT ,"ﾊﾟﾗﾒｰﾀ","ｹｯｶ"
230 :
240 ' -- ﾊﾟﾗﾒｰﾀ ﾊｶｲ
250 P$="ABC"
260 A$=USR(P$)
270 LPRINT "USR(P$)     ABC",P$
280 :
290 ' -- ﾊﾟﾗﾒｰﾀ ﾎｿﾞﾝ
300 P$="ABC"
310 A$=USR(P$+"")
320 LPRINT "USR(P$+";C$;C$;") ABC",P$
330 END
```

入っているデータがイコールの前に書かれた変数に代入されます。

コール時のレジスタの値は情報を知らせるためだけの役割なので，リターン時にどんな値が入っていてもかまいません(レジスタの保存は考える必要はない)。

実際にUSR命令を使用する場合のパラメータは，整数か文字列がほとんどだと思われます。ここでも整数パラメータでの実例をあげておきます(**リスト2-6**)。このプログラムは，特殊キーが入力されていたら-1 (TRUE) を，入力されてなかったら0 (FALSE) を返すものです。入力パラメータは**表2-9**のとおりです。この例ではファンクション・キー (テンキー，F1～F5のキーなど)が押されているかどうか調べています。

**リスト2-6** No. 1

```
10 '
20 ' USR ｶﾂﾖｳ ﾚｲ
30 '
40 CLEAR &HBFFF:CLS:PRINT "Hit any key!"
50 GOSUB 130
60 DEF USR=&HC000
70 P%=7              'FUNCTION
80 A=USR(P%)
90 LOCATE 12,8
100 IF A THEN PRINT "FUNCTION Key" ELSE PRINT SPC(20)
110 GOTO 80
120 '--- ｷｶｲｺﾞ  ｶｷｺﾐ
130 AD=&HC000
140 FOR I=0 TO 999
150   READ D$:IF D$="END" THEN 180
160   POKE AD+I,VAL("&H"+D$)
170 NEXT I
180 RETURN
190 :
200 DATA FE,02          :'       CP     02H
210 DATA 20,0B          :'       JR     NZ,ERR
220 DATA 23             :'       INC    HL
230 DATA 7E             :'       LD     A,(HL)
240 DATA B7             :'       OR     A
250 DATA 20,06          :'       JR     NZ,ERR
260 DATA 2B             :'       DEC    HL
270 DATA 7E             :'       LD     A,(HL)
280 DATA FE,08          :'       CP     08H
290 DATA 38,02          :'       JR     C,P1
300 DATA DD,E9          :'ERR    JP     (IX)
310 DATA 47             :'P1     LD     B,A
320 DATA 3E,01          :'       LD     A,1
```

No. 2

```
330 DATA 04          :'       INC   B
340 DATA 05          :'LOOP   DEC   B
350 DATA 28,03       :'       JR    Z,P2
360 DATA 87          :'       ADD   A,A
370 DATA 18,FA       :'       JR    LOOP
380 DATA 4F          :'P2     LD    C,A
390 DATA 3E,02       :'       LD    A,2
400 DATA CD,1B,00    :'       CALL  INKEY$
410 DATA A1          :'       AND   C
420 DATA 3E,00       :'       LD    A,00H
430 DATA 20,01       :'       JR    NZ,P3
440 DATA 3D          :'       DEC   A
450 DATA 77          :'P3     LD    (HL),A
460 DATA 23          :'       INC   HL
470 DATA 77          :'       LD    (HL),A
480 DATA C9          :'       RET
490 DATA END
500 :
510 END
```

表2 9 パラメータの値に対する機能

| P % | 機　　能 |
|---|---|
| 0 | CTRL キーが同時に押されているかを調べる |
| 1 | SHIFT キーが同時に押されているかを調べる |
| 2 | カナ キーが LOCK されているかどうかを調べる |
| 3 | CAPS キーが LOCK されているかどうかを調べる |
| 4 | GRAPH キーが同時に押されているかを調べる |
| 5 | リピート・データかどうかを調べる |
| 6 | データ・コードが有効かを調べる |
| 7 | テンキー，TV キー，カセット・キー，F1～F5 が押されたかを調べる |

USR関数でのマシン語プログラムの手順はおよそ次のようになります。

❶ FACから入力パラメータの値を取り出す。

❷パラメータが適正値かどうか調べ規定外の場合はエラー処理へジャンプする。IXレジスタにエラー処理アドレスが入っているのでJP (IX) とする。このときスタック・ポインタやレジスタの保存は気にしなくてよい。

❸パラメータから目的の値を計算する。または，目的の処理を行う。

❹ FACに出力パラメータをセットして戻る。

この例の場合，入力パラメータは0～7なので，それ以外の時はエラー処理へジャンプさせています。

単精度・倍精度実数をUSR関数の入力パラメータとして使い，マシン語ルーチンで実数を扱いたい場合は，**2-5-3**の浮動小数点演算ルーチンを利用すると良いでしょう。

# 2-8 HuBASICの拡張

HuBASICの拡張といっても，これだけ強力なBASICにいまさら手を加えることにどれほどの意義があるか疑問ですが，USR命令ではなく意味のある名前をつけたいときなどには有効でしょう。

## 2-8-1 ジャンプ・テーブルの解析

HuBASICの内部を解析するには，まずどこのアドレスで何の処理を行っているかを調べなければなりません。もっとも簡単な方法としては，**2-3**で説明した予約語テーブルからそれに対応するジャンプ・テーブルを探すことです。これは予約語テーブルのすぐ近くにあります。また，予約語テーブルと同様，〔通常のコマンド，ステートメント〕，〔拡張コマンド，ステートメント〕，〔関数〕の3つの部分に分かれていて，バラバラに配置されています。**表2-10**がそのアドレス一覧です。

これをもとに，予約語とジャンプ先の対応を表示するプログラムが**リスト2-7**です。テープ・バージョン，ディスク・バージョンの各ジャンプ・テーブル先頭アドレスをJ（0）～J（2）にセットし，ADに予約語テーブルの先頭アドレスをセットします。

**表2-10** ジャンプ・テーブル・アドレス

| | テープ | ディスク |
|---|---|---|
| 通常の命令 | 2CDFH | 2D0DH |
| 拡張命令 | 2D9FH | 2E09H |
| 関数 | 7698H | 7EF5H |

リスト2-7

```
100 '
110 ' Jump Table     X1 HuBASIC (TAPE)
120 '
130 CLS:WIDTH 40
140 AD=&H28F6   'Word Table Top Address
150 DIM J(2)
160 J(0)=&H2CDF 'Jump Table
170 J(1)=&H2D9F 'Jump Table (Extension)
180 J(2)=&H7698 'Jump Table (Function)
190 LF=0        'PRINT OUT FLAG
200 LL=50       'PRINT LINE
210 SP$=SPACE$(10)
220 DEF FNH$(X)=RIGHT$("000"+HEX$(X),4)
230 MSG$="NO.  Word        JumpAD  StoreAD"
240 IF LF<>0 THEN LPRINT MSG$:LPRINT
250 PRINT MSG$
260 '
270 JMPAD=J(0):CT=1:J=0
280 REPEAT
290   GOSUB "MAIN"
300   CT=CT+1:JMPAD=JMPAD+2
310 UNTIL CT=256
320 END
330 '
340 LABEL       "MAIN"
350 IF D=&HFF THEN J=J+1:JMPAD=J(J)
360 WD$=""
370   D=PEEK(AD):AD=AD+1
380   IF D=&H80 THEN WD$="???":GOTO 440
390   IF D=&HFF THEN WD$="   ":GOTO 440
400   XD=(D AND &H7F)
410   WD$=WD$+CHR$(XD)
420 IF D<&H80 THEN 370
430
440 IF LF<>0 THEN LPRINT USING "###  ";CT;
450 PRINT USING "###  ";CT;
460 P$=LEFT$(WD$+SP$,10)+"-> "
470 IF CT>96 AND CT<128 THEN 500
480   P$=P$+FNH$(PEEK(JMPAD)+PEEK(JMPAD+1)*256)
490   P$=P$+"   ["+FNH$(JMPAD)+"]"
500 PRINT P$
510 IF LF=0 THEN 560
520   LPRINT P$
530   IF (CT MOD LL)<>0 THEN 560
540   LPRINT CHR$(12)   'Top Feed
550   LPRINT MSG$:LPRINT
560 RETURN
```

これで表示されるストア・アドレス〔Store AD〕とは，その命令のジャンプ・アドレスが格納されているアドレスです。この表からおよそ，どこで何の処理をしているかがわかります（表2-11）。

**表2-11 ジャンプ・アドレス（テープ・バージョン）**

| NO. | Word | JumpAD | StoreAD |
|---|---|---|---|
| 1 | GOTO | -> 3456 | [2CDF] |
| 2 | GOSUB | -> 324D | [2CE1] |
| 3 | GO | -> 3448 | [2CE3] |
| 4 | RUN | -> 1795 | [2CE5] |
| 5 | RETURN | -> 3149 | [2CE7] |
| 6 | RESTORE | -> 2768 | [2CE9] |
| 7 | RESUME | -> 3348 | [2CEB] |
| 8 | LIST | -> 68F7 | [2CED] |
| 9 | LLIST | -> 68F2 | [2CEF] |
| 10 | DELETE | -> 2EF7 | [2CF1] |
| 11 | RENUM | -> 2F09 | [2CF3] |
| 12 | AUTO | -> 212F | [2CF5] |
| 13 | EDIT | -> 30A7 | [2CF7] |
| 14 | FOR | -> 17ED | [2CF9] |
| 15 | NEXT | -> 19AD | [2CFB] |
| 16 | PRINT | -> 1B0C | [2CFD] |
| 17 | LPRINT | -> 1B04 | [2CFF] |
| 18 | INPUT | -> 2335 | [2D01] |
| 19 | LINPUT | -> 22A6 | [2D03] |
| 20 | IF | -> 34E0 | [2D05] |
| 21 | DATA | -> 2E39 | [2D07] |
| 22 | READ | -> 279F | [2D09] |
| 23 | DIM | -> 879F | [2D0B] |
| 24 | REM | -> 15B7 | [2D0D] |
| 25 | END | -> 2128 | [2D0F] |
| 26 | STOP | -> 1F99 | [2D11] |
| 27 | CONT | -> 1FF3 | [2D13] |
| 28 | CLS | -> 3C82 | [2D15] |
| 29 | CLEAR | -> 2ECA | [2D17] |
| 30 | ON | -> 33DA | [2D19] |
| 31 | LET | -> 1657 | [2D1B] |
| 32 | NEW | -> 21A0 | [2D1D] |
| 33 | POKE | -> 3632 | [2D1F] |
| 34 | OFF | -> 205A | [2D21] |
| 35 | WHILE | -> 317D | [2D23] |
| 36 | WEND | -> 31B0 | [2D25] |
| 37 | REPEAT | -> 3222 | [2D27] |
| 38 | UNTIL | -> 3101 | [2D29] |
| 39 | ??? | -> 205A | [2D2B] |
| 40 | ??? | -> 205A | [2D2D] |
| 41 | ??? | -> 205A | [2D2F] |
| 42 | TRON | -> 2254 | [2D31] |
| 43 | TROFF | -> 2255 | [2D33] |
| 44 | ??? | -> 205A | [2D35] |
| 45 | ??? | -> 205A | [2D37] |
| 46 | ??? | -> 205A | [2D39] |
| 47 | DEFINT | -> 3574 | [2D3B] |
| 48 | DEFSNG | -> 357A | [2D3D] |
| 49 | DEFDBL | -> 357D | [2D3F] |
| 50 | DEFSTR | -> 3577 | [2D41] |
| 51 | DEF | -> 26AE | [2D43] |
| 52 | ??? | -> 205A | [2D45] |
| 53 | LOAD | -> 6AAE | [2D47] |
| 54 | SAVE | -> 6C04 | [2D49] |
| 55 | MERGE | -> 6B55 | [2D4B] |
| 56 | CHAIN | -> 6ADD | [2D4D] |
| 57 | CONSOLE | -> 38D9 | [2D4F] |
| 58 | WIDTH | -> 3694 | [2D51] |
| 59 | OUT | -> 2EBC | [2D53] |
| 60 | SEARCH | -> 6871 | [2D55] |
| 61 | WAIT | -> 35BD | [2D57] |
| 62 | PAUSE | -> 1B8F | [2D59] |
| 63 | WRITE | -> 1AA5 | [2D5B] |
| 64 | SWAP | -> 2E4F | [2D5D] |
| 65 | ERASE | -> 205A | [2D5F] |
| 66 | ERROR | -> 2061 | [2D61] |
| 67 | ELSE | -> 15B7 | [2D63] |
| 68 | CALL | -> 2DFD | [2D65] |
| 69 | MON | -> 0FE2 | [2D67] |
| 70 | LOCATE | -> 366B | [2D69] |
| 71 | SCREEN | -> 3BFD | [2D6B] |
| 72 | KEY | -> 3A32 | [2D6D] |
| 73 | ??? | -> 205A | [2D6F] |
| 74 | ??? | -> 205A | [2D71] |
| 75 | LABEL | -> 2E39 | [2D73] |
| 76 | RANDOMIZE | -> 2E1F | [2D75] |
| 77 | OPTION | -> 2272 | [2D77] |
| 78 | LINE | -> 3D3C | [2D79] |
| 79 | OPEN | -> 6CE7 | [2D7B] |
| 80 | CLOSE | -> 6C77 | [2D7D] |
| 81 | SIZE | -> 205A | [2D7F] |
| 82 | FIELD | -> 683D | [2D81] |
| 83 | GET | -> 4C3F | [2D83] |
| 84 | PUT | -> 4E1A | [2D85] |
| 85 | SET | -> 683D | [2D87] |
| 86 | FILES | -> 6E38 | [2D89] |

| | | | | |
|---|---|---|---|---|
| 87 | LFILES | -> | 6E37 | [2D8B] |
| 88 | DEVICE | -> | 6811 | [2D8D] |
| 89 | NAME | -> | 205A | [2D8F] |
| 90 | KILL | -> | 6844 | [2D91] |
| 91 | LSET | -> | 205A | [2D93] |
| 92 | RSET | -> | 205A | [2D95] |
| 93 | INIT | -> | 6E07 | [2D97] |
| 94 | VDIM | -> | 205A | [2D99] |
| 95 | MAXFILES | -> | 21E3 | [2D9B] |
| 96 | ??? | -> | 205A | [2D9D] |
| 97 | TO | -> | | |
| 98 | STEP | -> | | |
| 99 | THEN | -> | | |
| 100 | USING | -> | | |
| 101 | SUB | -> | | |
| 102 | BASE | -> | | |
| 103 | TAB | -> | | |
| 104 | SPC | -> | | |
| 105 | EQV | -> | | |
| 106 | IMP | -> | | |
| 107 | XOR | -> | | |
| 108 | OR | -> | | |
| 109 | AND | -> | | |
| 110 | NOT | -> | | |
| 111 | >< | -> | | |
| 112 | <> | -> | | |
| 113 | =< | -> | | |
| 114 | <= | -> | | |
| 115 | => | -> | | |
| 116 | >= | -> | | |
| 117 | = | -> | | |
| 118 | > | -> | | |
| 119 | < | -> | | |
| 120 | + | -> | | |
| 121 | - | -> | | |
| 122 | MOD | -> | | |
| 123 | ¥ | -> | | |
| 124 | / | -> | | |
| 125 | * | -> | | |
| 126 | ^ | -> | | |
| 127 | | -> | | |
| 128 | WINDOW | -> | 3772 | [2D9F] |
| 129 | PSET | -> | 3B4D | [2DA1] |
| 130 | PRESET | -> | 3B42 | [2DA3] |
| 131 | COLOR | -> | 3AA5 | [2DA5] |
| 132 | CIRCLE | -> | 41B2 | [2DA7] |
| 133 | POLY | -> | 41B5 | [2DA9] |
| 134 | PAINT | -> | 47C1 | [2DAB] |
| 135 | ??? | -> | 205A | [2DAD] |
| 136 | POSITION | -> | 4F39 | [2DAF] |
| 137 | PATTERN | -> | 4F45 | [2DB1] |
| 138 | HCOPY | -> | 50AC | [2DB3] |
| 139 | PLAY | -> | 44E7 | [2DB5] |
| 140 | SOUND | -> | 475D | [2DB7] |
| 141 | BEEP | -> | 477A | [2DB9] |
| 142 | PRW | -> | 3B23 | [2DBB] |
| 143 | PALET | -> | 3AE6 | [2DBD] |
| 144 | LAYER | -> | 4AF9 | [2DBF] |
| 145 | CANVAS | -> | 4AE5 | [2DC1] |
| 146 | CREV | -> | 416A | [2DC3] |
| 147 | CFLASH | -> | 4181 | [2DC5] |
| 148 | CGEN | -> | 4191 | [2DC7] |
| 149 | CSIZE | -> | 41A1 | [2DC9] |
| 150 | EJECT | -> | 4A86 | [2DCB] |
| 151 | CSTOP | -> | 4A88 | [2DCD] |
| 152 | FAST | -> | 4A8B | [2DCF] |
| 153 | REW | -> | 4A8E | [2DD1] |
| 154 | APSS | -> | 4A98 | [2DD3] |
| 155 | TVPW | -> | 52DA | [2DD5] |
| 156 | CHANNEL | -> | 52F3 | [2DD7] |
| 157 | VOL | -> | 52FF | [2DD9] |
| 158 | CRT | -> | 5331 | [2DDB] |
| 159 | SCROLL | -> | 4BEA | [2DDD] |
| 160 | EFFECT | -> | 4C2A | [2DDF] |
| 161 | GRAPH | -> | 3BFD | [2DE1] |
| 162 | MUSIC | -> | 44E7 | [2DE3] |
| 163 | TEMPO | -> | 44E7 | [2DE5] |
| 164 | CURSOR | -> | 366B | [2DE7] |
| 165 | VERIFY | -> | 69EA | [2DE9] |
| 166 | CLR | -> | 21EE | [2DEB] |
| 167 | LIMIT | -> | 2ECD | [2DED] |
| 168 | KLIST | -> | 393D | [2DEF] |
| 169 | ASK | -> | 52A1 | [2DF1] |
| 170 | KBUF | -> | 4C2D | [2DF3] |
| 171 | CLICK | -> | 52CB | [2DF5] |
| 172 | BOOT | -> | 5293 | [2DF7] |
| 173 | DEVI$ | -> | 6719 | [2DF9] |
| 174 | DEVO$ | -> | 676D | [2DFB] |
| 175 | | -> | 4DCD | [2DFD] |
| 176 | INT | -> | 8A03 | [7698] |
| 177 | ABS | -> | 89FB | [769A] |
| 178 | SIN | -> | 8BCC | [769C] |
| 179 | COS | -> | 8BB2 | [769E] |
| 180 | TAN | -> | 8CC6 | [76A0] |
| 181 | LOG | -> | 8F84 | [76A2] |
| 182 | EXP | -> | 8E6C | [76A4] |
| 183 | SQR | -> | 8A5C | [76A6] |
| 184 | RND | -> | 8E3A | [76A8] |
| 185 | PEEK | -> | 8E31 | [76AA] |
| 186 | ATN | -> | 8AEA | [76AC] |
| 187 | SGN | -> | 8DD7 | [76AE] |
| 188 | FRAC | -> | 8A3C | [76B0] |
| 189 | FIX | -> | 8A33 | [76B2] |
| 190 | PAI | -> | 8E08 | [76B4] |
| 191 | RAD | -> | 8E00 | [76B6] |
| 192 | INP | -> | 8E24 | [76B8] |
| 193 | CDBL | -> | 53D9 | [76BA] |
| 194 | CSNG | -> | 53F6 | [76BC] |
| 195 | CINT | -> | 5A50 | [76BE] |
| 196 | DSKF | -> | 680E | [76C0] |
| 197 | EOF | -> | 6CBF | [76C2] |
| 198 | FPOS | -> | 680E | [76C4] |

| | | | | |
|---|---|---|---|---|
| 199 | LOC | -> | 680E | [76C6] |
| 200 | LOF | -> | 680E | [76C8] |
| 201 | POS | -> | 67E7 | [76CA] |
| 202 | FAC | -> | 8A96 | [76CC] |
| 203 | SUM | -> | 8A71 | [76CE] |
| 204 | FRE | -> | 7C24 | [76D0] |
| 205 | LPOS | -> | 67DF | [76D2] |
| 206 | STICK | -> | 7C6F | [76D4] |
| 207 | STRIG | -> | 7C53 | [76D6] |
| 208 | CHR$ | -> | 7CD0 | [76D8] |
| 209 | STR$ | -> | 7E9B | [76DA] |
| 210 | HEX$ | -> | 7D4A | [76DC] |
| 211 | OCT$ | -> | 7D34 | [76DE] |
| 212 | BIN$ | -> | 7D3F | [76E0] |
| 213 | MKI$ | -> | 7DC6 | [76E2] |
| 214 | MKS$ | -> | 7DCB | [76E4] |
| 215 | MKD$ | -> | 7DD0 | [76E6] |
| 216 | SPACE$ | -> | 7DDC | [76E8] |
| 217 | CGPAT$ | -> | 7E02 | [76EA] |
| 218 | KANJI$ | -> | 7E26 | [76EC] |
| 219 | ASC | -> | 7EBE | [76EE] |
| 220 | LEN | -> | 7ECA | [76F0] |
| 221 | VAL | -> | 7ED2 | [76F2] |
| 222 | CVS | -> | 7EEC | [76F4] |
| 223 | CVD | -> | 7EF0 | [76F6] |
| 224 | CVI | -> | 7EE8 | [76F8] |
| 225 | ??? | -> | 205A | [76FA] |
| 226 | ??? | -> | 205A | [76FC] |
| 227 | ERR | -> | 7C42 | [76FE] |
| 228 | ERL | -> | 7CB3 | [7700] |
| 229 | CSRLIN | -> | 7C3A | [7702] |
| 230 | STRPTR | -> | 7C47 | [7704] |
| 231 | DTL | -> | 7C4D | [7706] |
| 232 | ??? | -> | 205A | [7708] |
| 233 | ??? | -> | 205A | [770A] |
| 234 | LEFT$ | -> | 7F61 | [770C] |
| 235 | RIGHT$ | -> | 7F79 | [770E] |
| 236 | MID$ | -> | 7F97 | [7710] |
| 237 | INKEY$ | -> | 803D | [7712] |
| 238 | INSTR | -> | 818E | [7714] |
| 239 | HEXCHR$ | -> | 821B | [7716] |
| 240 | MEM$ | -> | 807D | [7718] |
| 241 | SCRN$ | -> | 80B4 | [771A] |
| 242 | VARPTR | -> | 7F07 | [771C] |
| 243 | STRING$ | -> | 8135 | [771E] |
| 244 | TIME | -> | 800C | [7720] |
| 245 | DAY$ | -> | 7FEC | [7722] |
| 246 | DATE$ | -> | 7FFD | [7724] |
| 247 | FN | -> | 8820 | [7726] |
| 248 | USR | -> | 82E1 | [7728] |
| 249 | ??? | -> | 205A | [772A] |
| 250 | ??? | -> | 205A | [772C] |
| 251 | ATTR$ | -> | 205A | [772E] |
| 252 | POINT | -> | 810E | [7730] |
| 253 | CHARACTER$ | -> | 80A9 | [7732] |
| 254 | CMT | -> | 7F19 | [7734] |
| 255 | MIRROR$ | -> | 82BA | [7736] |

# 2-8-2 既存命令の変更

BASIC の拡張には,

①既存する命令の変更

②新しいコマンド・ステートメントの追加

③新しい関数の追加…パラメータの受け渡しが必要

が考えられますが，まずはすでにある命令をより使い安くしてみましょう。

X1 Turbo の BASIC では AUTO 命令が拡張されていて，AUTO *というのがあります。これは，自動的に行番号を出力するとともに REM の省略形である (') も続けて出力するものです。ですから，REM 文の多いリストの入力や BASIC のエディタを他の言語のソース入力用として利用する場合に有効です（そのかわり EDIT 命令で先頭の' が表示されなくなります)。

次からの説明の中で出てくるアドレスはテープ・バージョンのものです。ディスク・バージョンで行いたい方は説明を参考にしながら自分で解析してください。明確にディスク・バージョンのアドレスがわかるときはカッコ内で示します。

まず AUTO 命令の処理先は 212FH (2150H) であることが**表2-11** よりわかります。AUTO 命令の文字列を見つけるとここがコールされます。

このときのレジスタの使われ方ですが，特に意味のあるのは HL レジスタで "AUTO" の次の文字を示すアドレスが入っています。

**図2-20**は AUTO 命令処理部分の逆アセルブルです。212FH には『LD DE, 000AH』があったのでこれをコール命令に変えます。そしてコール先では, HL が文字列のポインタですから HL の示すメモリに，*があるかどうかチェックします。

解析を進めていくと，AUTO 命令の実行部分は，30ADH (30DBH) 以降であることがわかりました。

30F0H (3112H) に "CALL 04BAH" とあり，これが行番

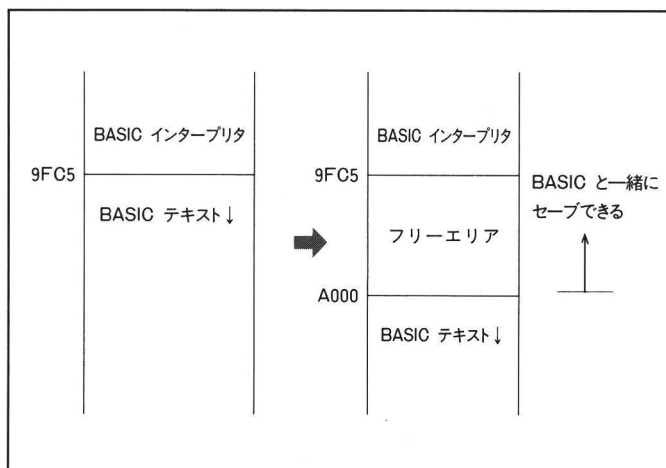
号の次にスペースを表示する部分です。このあたりを書き換えて（'）を表示させるようにします。

さて，こういったプログラムをどこへ置くかということですが，もっとも簡単で確実な方法としては，BASICテキストのポインタを書き換えて，BASICテキストをもっと上のアドレスにずらして空いた部分を使用する方法です（**図2-21**）。この方法なら拡張部分もBASICインタープリタといっしょにセーブできます。直接ポインタを書き換えても良いのですがBASICのNEW ON命令を使った方が簡単でしょう。

**図2-20 AUTO命令逆アセンブル・リスト**

```
Addr    Mnemonic

212F    LD      DE,000AH  ; DE←スタート番号
2132    LD      BC,000AH  ; BC←増分
2135    JR      Z,2162H   ; AUTO⏎のときジャンプ
2137    CP      2CH       ; ",”かどうか調べて
2139    JR      NZ,2144H  ; 違うとき2144Hへジャンプ
213B    CALL    305AH     ; 行番号取り込み
 .       .       .
 .       .       .
 .       .       .
2144    CALL    7A05H     ; スペースでないところまでHLを増やす
```

**図2-21 NEW ON &HA000を実行**

**リスト2-8**がAUTO拡張用のプログラム（テープ・バージョン）ですが，これを打ち込む前に必ずNEW ON &HA000を実行しておいてください。

このプログラムを走らせた後，モニタへ飛んで，

| ＊S0000　9FFF　　0000：〈ファイル名〉⏎ |
|---|

とするとテープに拡張されたHuBASICがセーブされます。AUTO命令の変更以外は普通のHuBASICとして使えます。

リスト2-8

```
10 '
20 ' AUTO ｺﾏﾝﾄﾞ ｶｸﾁｮｳ (TAPE)
30 '
40 AD=&H9FC5
50 FOR I=0 TO 999
60   READ D$:IF D$="END" THEN 100
70   POKE AD+I,VAL("&H"+D$)
80 NEXT I
90 :
100 DATA CD,BA,04        :'      CALL  SPPRT
110 DATA 3E,27           :'      LD    A,27H (')
120 DATA C3,BC,04        :'      JP    ACPRT
130 '--- 9FCDH ---
140 DATA D1              :'      POP   DE
150 DATA 3A,E1,30        :'      LD    A,(30E1H)
160 DATA FE,BA           :'      CP    BAH
170 DATA 28,06           :'      JR    Z,P1
180 DATA 1A              :'      LD    A,(DE)
190 DATA FE,27           :'      CP    27H
200 DATA 20,01           :'      JR    NZ,P1
210 DATA 13              :'      INC   DE
220 DATA 06,00           :'P1    LD    B,00H
230 DATA C3,EE,30        :'      JP    30EEH
240 '--- 9FE0H ---
250 DATA 11,0A,00        :'      LD    DE,10
260 DATA CD,05,7A        :'      CALL  SPPASS
270 DATA FE,2A           :'      CP    '*'
280 DATA 20,0i           :'      JR    NZ,P2
290 DATA 23              :'      INC   HL
300 DATA E5              :'P2    PUSH  HL
310 DATA 21,BA,04        :'      LD    HL,04BAH
320 DATA FE,2A           :'      CP    '*'
330 DATA 20,03           :'      JR    NZ,P3
340 DATA 21,C5,9F        :'      LD    HL,9FC5H
350 DATA 22,E1,30        :'P3    LD    (30E1H),HL
360 DATA E1              :'      POP   HL
370 DATA CD,05,7A        :'      CALL  SPPASS
380 DATA B7              :'      OR    A
390 DATA C9              :'      RET
400 DATA END
410 :
420 POKE &H212F,&HCD,&HE0,&H9F
430 POKE &H30EB,&HC3,&HCD,&H9F
440 END
```

## 2-8-3 新しい命令の追加

新しい命令や関数を作る場合のアイデアの1つとして重複命令を利用する方法が考えられます。HuBASICにはまったく同じ動作をする命令で重複しているものがあります。LOCATEとCURSORがそのよい例で，まったく同じ処理に飛んでいます。あまり使用しない方の予約語を自分の好きなスペルに書き換え，それに対応しているジャンプ・テーブルを書き換えれば，比較的簡単に新しい命令が追加できます。また，あまり使用しない命令を削除してもよいでしょう。

このとき注意することは，予約語の長さを変えないことです。どうしても，予約語の長さを変えたいときは予約語テーブル全体をうまく移動させる必要があります。

さて，新しい命令の追加例としてはLPOS関数の変わりに，IPL ROMの内容を読むPEEK関数を作ってみましょう。関数名は『IPEK』とします。

コマンドやステートメントと関数では，ジャンプ・テーブルから飛んできたときのレジスタの値の意味や，リターンするときの条件が**表2-12**のように異なります。関数の場合，パラメータの受け渡しが重要となりますが，それらはすべてFACを通じて行われます。

**表2-12 HL レジスタとリターン時の条件**

| | コマンド，ステートメント | 関　数 |
|---|---|---|
| コール時のHL レジスタの意味 | コマンド，ステートメントにつづく文字列が格納されているアドレス | パラメータの格納してある FAC先頭アドレス |
| リターン時の条件 | HL の値を次のコマンド，ステートメントの先頭アドレスまで移動させておかなければならない。HLレジスタは保存すること。 | 9CF8H（ディスク：A5DBH）にリターン・パラメータのデータの種類を，FACにデータを，セットしたら全レジスタを破壊してもかまわない。 |

STICK関数を解析していったところ，FACから2バイトの整数データを取り出すサブルーチンが見つかったので，これを利用します(**表2-13**)。ところが，このサブルーチンではHLを破壊しています。FACにリターン・パラメータのデータをセットするまではそのアドレスを壊すわけにはいかないので退避させておきます。

**表2-13 FACに関するサブルーチン・エントリー**

| アドレス | 内容 |
|---|---|
| 7D6DH<br>(85CAH) | FACから2バイト整数データを取り出す<br>●入力　HL = FAC先頭アドレス<br>●出力　HL =データ，A =下位データ |
| 7C31H<br>(848EH) | FACに2バイト整数データをセットし，データの種類を02Hにしてリターン<br>●入力　HL = FAC先頭アドレス，DE =データ |
| 7C3DH<br>(849AH) | FACに1バイト整数データをセットし，データの種類を02Hにしてリターン<br>●入力　HL = FAC先頭アドレス，A =データ |

次にエラー判断を行います。IPL ROMの有効なアドレスは0000H～0FFFHですからそれ以上の値がきたら，エラー処理ルーチンへジャンプさせます。エラー処理ルーチンではスタック・ポインタも初期化しているので，スタックを気にせず，レジスタの値もそのままでジャンプしてかまいません。

各種エラーエントリーでは，Aレジスタにエラー番号をセットして，エラーメッセージ表示へ行くのですが，ここでちょっとしたプログラム・テクニックが使用されているので紹介します。

エラーエントリーの所を2057Hから逆アセンブルすると，

```
アドレス    データ          ニモニック
  2057    3E 16        LD    A, 16H
  2059    21 3E 22     LD    HL, 223EH
  205C    21 3E 05     LD    HL, 053EH
  205F    18 03        JR    2064H (エラー処理へ)
```

となり，Aレジスタに16Hが入ります。しかし，205AHから逆アセンブルすると，

| アドレス | データ | ニモニック | |
|---|---|---|---|
| 205A | 3E 22 | LD | A，22H |
| 205C | 21 3E 05 | LD | HL，053EH |
| 205F | 18 03 | JR | 2064H（エラー表示へ） |

となり，ジャンプしてくるアドレスにより，Aレジスタに入る値（この場合エラー番号）を選べるようになっています。プログラムを小さくするテクニックのひとつです。**表2-14**に各種エラーエントリーをまとめておきました。

さて，本題に戻りましょう。エラー判断をすませた後はIPL ROMから値を読み，それをFACに書き込めばよいわけです。

FACに整数データを書き込み，データの種類(整数＝02H)を9CF8HにセットしてくれるルーチンもすでにHuBASIC内にあるのでそのルーチンを利用します(**表2-13**)。ちなみに，そのルーチンでは次のような処理を行っています。

```
アドレス  データ      ニモニック
  7C3D     5F          LD  E, A  ┐ DEに1バイト
  7C3E     16 00       LD  D, 00 ┘ を入れる
  7C40     18 EF       JR  7C31H
                              │
  ┌───────────────────────────┘
  ▼
  7C31     73          LD  (HL), E ┐ FACにデー
  7C32     23          INC  HL     │ タをセット
  7C33     72          LD  (HL), D ┘
  7C34     3E 02       LD  A, 02H ─┐ データの種
  7C36     32 F8 9C    LD (9CF8H), A┘ 類をセット
  7C39     C9          RET
```

**リスト2-9**がIPEK関数拡張プログラムです。さきほどと同様，必ずNEW ON &HA000をダイレクトに実行してから打ち込んでください。テープへのセーブの方法も前と同じです。

表2-14 エラーエントリーアドレス

( )内はディスク・バージョン

| アドレス | エラー番号 | 種　　類 |
| --- | --- | --- |
| 200D (201F) | 7 | Out of memory（カセット・ストップ，CLR 実行） |
| 2015 (2027) | 7 | Out of memory |
| 2018 (202A) | 1 | NEXT without FOR |
| 201B (202D) | 8 | Undefined label |
| 201E (2030) | 10 | Duplicate Definition |
| 2021 (2033) | 9 | Subscript out of range |
| 2024 (2036) | 20 | RESUME without error |
| 2027 (2039) | 32 | WHILE without WEND |
| 202A (203C) | 33 | WEND without WHILE |
| 202D (203F) | 26 | UNTIL without REPEAT |
| 2030 (2042) | 3 | RETURN without GOSUB |
| 2033 (2045) | 17 | Can't continue |
| 2036 (2048) | 15 | String too long |
| 2039 (204B) | 63 | Unprintable error |
| 203C (204E) | 30 | Bad file mode |
| 203F (2051) | 19 | No RESUME |
| 2042 (2054) | 23 | Line buffer overflow |
| 2045 (2057) | 11 | Divison by zero |
| 2048 (205A) | 6 | Overflow |
| 204B (205D) | 16 | Too complex |
| 204E (2060) | 13 | Type mismatch |
| 2051 (2063) | 35 | FOR without NEXT |
| 2054 (2066) | 2 | Syntax error |
| 2057 (2069) | 22 | Missing operand |
| 205A (206C) | 34 | Reserved feature |
| 205D (206F) | 5 | Illegal function call |

リスト2-9

```
10 '
20 ' IPEK ｶﾝｽｳ ｶｸﾁｮｳ (TAPE)
30 '
40 AD=&H9FE0
50 FOR I=0 TO 999
60   READ D$:IF D$="END" THEN 100
70   POKE AD+I,VAL("&H"+D$)
80 NEXT I
90 :
100 DATA E5             :'PUSH  HL
110 DATA CD,6D,7D       :'CALL  7D6D
120 DATA 7C             :'LD    A,H
130 DATA FE,10          :'CP    10H
140 DATA D2,5D,20       :'JP    NC,205DH
150 DATA 06,1D          :'LD    B,1DH
160 DATA ED,79          :'OUT   (C),A
170 DATA 7E             :'LD    A,(HL)
180 DATA 06,1E          :'LD    B,1EH
190 DATA ED,79          :'OUT   (C),A
200 DATA E1             :'POP   HL
210 DATA C3,3D,7C       :'JP    7C3DH
220 DATA END
230 :
240 '-- Word "IPEK"
250 POKE &H2C0A,&H49,&H50,&H45,&HCB
260 '-- Jump Address
270 POKE &H76D2,&HE0,&H9F
280 END
```

# 2-9 モニタの拡張

HuBASICには,スクリーン・エディット可能で,便利なF(ファインド) コマンドをも備えているモニタが付いています。しかし,マシン語だけのプログラムを打ち込むときはBASICは不要ですし,またもう少し機能が欲しいところです。ここでは,モニタだけのテープをつくる方法とモニタの解析・拡張例を紹介します。

## 2-9-1 モニタの切り放し

Huモニタは HuBASIC中のサブルーチンを使用することなく独立しているために切り放しが可能となります。ただし,IOCSに多く依存しているため,当然IOCS部分も含めて切り放します。切り放しのときに変更しなければならない箇所が2つあります。

第1に,切り放したHuモニタをテープからロードし,起動したときにモニタのコマンド待ちにジャンプさせなければなりません。変更前ではBASICのコマンド待ちに飛ぶようにつくられているため,そのままでは暴走してしまいます。

IOCS部分で画面やキー入力の初期化が行われた後BASICへジャンプさせるジャンプ・アドレスは,012BHと012CHにおかれています。ここをモニタのスタート・アドレスである1000Hに書き換えます。

第2にはBASICへ戻るためのRコマンドを使えなくしなければなりません。そうしないと,うっかりRコマンドを使ってしまって暴走,ということになりかねないからです。Rコマンドを封じ込めるもっとも簡単な方法としては,コマンド・テーブルの中の『R』を,Rコマンドより優先順位の高いDコマンドなどに変えてしまえばよいのです。Huモニタの

コマンドとジャンプ・テーブルは1034H～1060Hにあり，Rコマンドは1052Hにあります。

以上にHuモニタ切り放しの手順を箇条書きにします。たへん簡単ですので，1度ためしてみてください。

❶ HuBASICをロードし，MONでモニタに移る。
❷ Mコマンドで，012BHのC2Hを00H，012CHの14Hを10Hに変える。
❸同様に1052Hの52Hを44H（"R"を"D"）に変える。
❹ Newテープをセットして，＊S0000 147F 0000：MON ⏎としてテープにセーブする。

これでモニタ・オンリーのテープができあがります。これをIPLから起動させるとすぐにモニタのコマンド待ちになります。

## 2-9-2 コマンドの拡張

Huモニタの機能がいかに強力だとはいえ，まだまだ不備な点はあるものです。たとえば，先に行ったようにモニタを切り放し，マシン語入力ツールとして使う場合，チェックサムの表示機能がどうしても欲しいところです。そこで，次のようなコマンドを追加してみましょう。

Cコマンド（CHECK SUM）
機能：ダンプ表示（Dコマンド）のときのチェックサムの方式を変更する。
C0 …1行8バイト・ダンプ表示
チェックサムは（8バイトの合計値）の下位から16進で2文字表示
C1 …1行8バイト・ダンプ表示
チェックサムは（8バイト合計＋アドレス）の下位2桁
C2 …1行8バイト・ダンプ表示
チェックサムは（8バイトの合計値）の下位3桁，

```
      64バイトごとに全合計値を4桁で表示
C3  …1行16バイト・ダンプ表
      チェックサムは縦横表示で下位2桁
C   …ASCII表示に戻し，チェックサムは表示しない。
```

さて，Huモニタのコマンド検出部(101D～1042)は次のようになっています。

```
         EXX
         LD    HL,1043H ; HL←コマンド・テーブ
                           ル・トップ
         LD    B,10      ;B←コマンド総数
LOOP:    CP    (HL)
         INC   HL
         JR    Z, EXEC  ; コマンドが見つかったの
                           でEXEC (103DH) へ
         INC   HL
         INC   HL
         DJNZ  L00P
         EXX
         RET             ; コマンドでないのでリターン
           ・
           ・
           ・
EXEC:    LD    E,(HL)
         INC   HL
         LD    D, (HL)   ;DE←ジャンプ・アドレス
         PUSH  DE        ; RETでジャンプするため
         EXX
         RET             ;DEの内容へジャンプ
```

つまり，Huモニタのコマンドは全部で10個でコマンド・テーブルは1043H～1060Hにあるということです。このテーブルは次のような構造になっています。

| アドレス | データ | ニモニック | 意味 |
|---|---|---|---|
| 1043 | 44 | DEFM 'D' | Dコマンドの処理先は |
| 1044 | 8B 11 | DEFW 118BH | 118BH |
| 1046 | 4D | DEFM 'M' | Mコマンドの処理先は |
| 1047 | 1D 12 | DEFW 121DH | 121DH |
| | | ・ | |
| | | ・ | |
| | | ・ | |

コマンドを増やすにはコマンド総数を増やし，アドレスの1061H以降にコマンドテーブルを追加していけば良いわけです。ただし，1061H～1069HはPコマンド処理に使われていますので，これを他のエリアへ移さなければなりません。モニタを切り放す場合は14C0H以降がフリーエリアとして使えます。

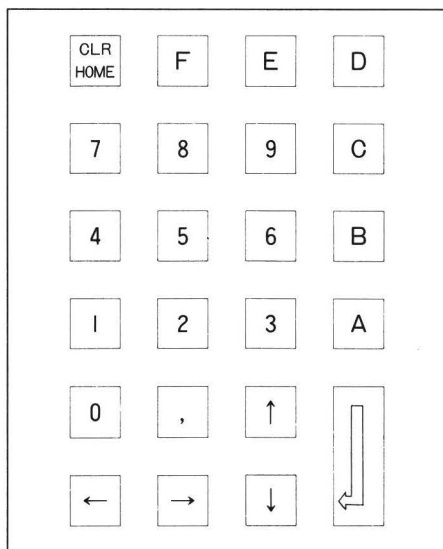
コマンド・テーブルの拡張をすませて，Cコマンドを入力したときにあるアドレスへジャンプするようにしたとします。そして，たとえばモニタキー入力待ちのときに＊C0⏎と入力した場合，ジャンプした時点でDEレジスタには入力文字列中の"C"の次の文字のアドレスが格納されています。よって，DEレジスタが示すメモリが"0"～"3"を調べて，どのチェックサム方式を指定したのかを判断できます。

Pコマンドの処理部分を他のエリアに移動させたことにより，9バイト空きますから，コマンドを3つまで増やすことができます。数値取り込みサブルーチンの使い方になれるために，もう2個コマンドを拡張しましょう。

ひとつは指定されたメモリを一定のデータで埋めるWコマンドで，もうひとつはテンキーを16進キーに変えるHコマンドです。

W コマンド (Write full)
・機能：スタートアドレスとエンドアドレスで指定された範囲のメモリすべてにデータを書き込む
・形式：＊スタートアドレス　エンドアドレス　データ
H コマンド (HEX KEY)
・機能：テンキーを16進キーに変える（**図2-22**）
・形式：＊ H ↵ を行うごとに切り換わる

**図2-22 16進キー配列**

| | | | |
|---|---|---|---|
| CLR HOME | F | E | D |
| 7 | 8 | 9 | C |
| 4 | 5 | 6 | B |
| 1 | 2 | 3 | A |
| 0 | , | ↑ | ↵ |
| ← | → | ↓ | |

Wコマンドでは『2バイト数値を3回取り出す』というモニタ内のサブルーチンを使っています。第3アドレスはHLレジスタに2バイトとして取り出されますが，下位であるLレジスタの値をデータとして使用します。このサブルーチンについては，次の**2-9-3**を参照してください。

**リスト2-10**がこれまで述べたモニタ拡張のためのプログラムです。モニタ切り放しも行っていますが，変更すればBASICとの同居も可能でしょう。新しいテープをセットして，RUNしてください。自動的にIPL起動のテープができます。

**リスト2 10** No. 1

```
100 ' Huﾓﾆﾀ ﾉ ｶｸﾁｮｳ (exMON)
110 :
120 CLEAR &HDFFF:AD=&HE000
130 FOR I=0 TO 999
140   READ D$:IF D$="END" THEN 220
150   P=1
160   FOR J=0 TO 15
170     D=VAL("&H"+MID$(D$,P,2)):POKE AD+I*16+J,D
180     P=P+3
190   NEXT J
200 NEXT I
210 :
220 POKE &H1022,13                'ｺﾏﾝﾄﾞ ﾉ ｶｽﾞ
230 POKE &H1061,&H43,&HFB,&H15  'CCOM
240 POKE &H1064,&H57,&HC9,&H14  'WCOM
250 POKE &H1067,&H48,&HD7,&H14  'HCOM
260 POKE &H104A,&HC0,&H14        'PCOM
270 POKE &H118C,&HBA,&H16        'DCOM
280 POKE &H11B0,&H67,&H16        'DCOM2
290 :
300 CALL &HE210
310 :
320 DATA 3A 72 14 EE 01 32 72 14 C9 CD A6 12 38 08 7D 12
330 DATA 13 0B 78 B1 20 F8 C9 21 ED 14 3A EC 14 B7 28 03
340 DATA 21 AA 02 22 9C 01 EE FF 32 EC 14 C9 00 F3 CD AA
350 DATA 02 D5 16 41 FE 2E 28 1C 14 FE 3D 28 17 14 FE 2B
360 DATA 28 12 14 FE 2D 28 0D 14 FE 2A 28 08 14 FE 2F 28
370 DATA 03 D1 FB C9 7A D1 FB C9 3E 3A CD 20 14 3E 20 CD
380 DATA 20 14 41 D5 11 5A 15 ED 53 58 15 1E 30 AF 00 00
390 DATA 57 7E 82 30 01 1C 57 7E E5 2A 58 15 86 77 23 22
400 DATA 58 15 E1 23 10 EB D5 11 6B 15 D1 7A CD 07 12 CD
410 DATA 8E 15 D1 E3 E1 C3 DE 11 5A 15 00 00 00 00 00 00
420 DATA 00 00 00 00 00 00 00 00 00 00 00 7B CD 20 14 3E
430 DATA 2E CD 20 14 C9 3A FA 15 3C 32 FA 15 B9 C9 CD 46
440 DATA 14 3E 2D CD 20 14 10 FB 3E 20 CD 20 14 C9 CD 75
450 DATA 15 C0 E5 C5 06 3A CD 7E 15 CD 46 14 06 06 3E 20
460 DATA CD 20 14 10 F9 41 0E 00 21 5A 15 7E CD 07 12 7E
470 DATA 81 4F 36 00 3E 20 CD 20 14 23 10 EF 3E 3A CD 20
480 DATA 14 3E 20 CD 20 14 79 CD 07 12 CD 46 14 AF 32 FA
490 DATA 15 C1 E1 C9 CD 75 15 C0 E5 D5 06 1F CD 7E 15 11
```

No. 2

```
500 DATA 00 00 06 40 2B 7E E5 6F 26 00 19 EB E1 10 F5 EB
510 DATA CD 02 12 AF 32 FA 15 D1 E1 C9 03 3E 08 32 A2 11
520 DATA 3E 11 32 47 15 32 4F 15 21 7F 00 22 9D 11 2E 00
530 DATA 22 2E 15 21 18 15 22 CA 11 3A 07 00 FE 50 20 05
540 DATA D5 CD 98 09 D1 1A FE 30 C8 FE 31 28 0F FE 32 28
550 DATA 12 FE 33 28 18 21 20 14 22 CA 11 C9 21 85 84 22
560 DATA 2E 15 C9 3E CD 32 47 15 21 D4 15 18 11 CD 8C 09
570 DATA 3E 10 32 A2 11 21 FF 00 22 9D 11 21 8E 15 22 50
580 DATA 15 3E CD 32 4F 15 C9 CD 4E 12 3A FA 15 B7 C0 3A
590 DATA A2 11 FE 10 C0 D5 11 7E 16 CD 2F 14 D1 C9 41 64
600 DATA 64 72 20 2B 30 20 2B 31 20 2B 32 20 2B 33 20 2B
610 DATA 34 20 2B 35 20 2B 36 20 2B 37 20 2B 38 20 2B 39
620 DATA 20 2B 41 20 2B 42 20 2B 43 20 2B 44 20 2B 45 20
630 DATA 2B 46 20 3A 53 75 6D 0D 3A 00 AF 32 FA 15 E5 21
640 DATA 5A 15 3E 11 36 00 23 3D 20 FA E1 CD 1F 11 C9 00
650 DATA 21 00 E0 11 C0 14 01 10 02 ED B0 3E 44 32 52 10
660 DATA 21 00 10 22 2B 01 11 39 E2 D9 01 D0 16 11 00 00
670 DATA 21 00 00 CD 6E 10 C3 00 10 3A 65 78 4D 4F 4E 00
680 DATA END
```

## 2-9-3 モニタ内ルーチンの解析

**表2-15, 2-16, 2-17**にモニタのサブルーチン，エントリー，ワーク・エリアを示します。モニタ内の解析および，自分でモニタを拡張するときの参考にしてください。

**表2-15 モニタ内サブルーチン**

| アドレス | 内容 |
|---|---|
| 画面表示サブルーチン I (P コマンドにより CRT，プリンタの出力切り換え可) | |
| 11AF | 1行メモリ・ダンプ<br>●「：アドレス＝XX XX XX XX XX XX XX XX/ ASCII ダンプ」形式で1行表示する。<br>●入力 HL …スタート・アドレス<br>DE …エンド・アドレス－1<br>●出力 HL …スタート・アドレス＋8<br>キャリーフラグ… 0のとき正常<br>1のとき (SHIFT ＋ BREAK を押した / アドレスが0000Hになった)<br>ゼロフラグ… 0のとき正常<br>1のとき SHIFT ＋ BREAK を押した<br>●レジスタ A, A', HL, B 以外保存 |
| 1202 | 16進4桁表示<br>●HLレジスタの値を16進4桁で表示。<br>●入力 HL…表示データ<br>●出力 なし<br>●レジスタ A, A' 以外保存 |
| 1207 | 16進2桁表示<br>●A レジスタの値を16進2桁で表示。<br>●入力 A …表示データ<br>●出力 なし<br>●レジスタ A, A' 以外保存 |
| 124A | “＝” を表示<br>●入力 なし<br>●レジスタ A, A' 以外保存 |

| アドレス | 内　　容 |
|---|---|
| 124E | “：”を表示<br>●入力　なし<br>●レジスタ　A, A′ 以外保存 |
| 1420 | 1文字表示<br>●A レジスタの値を ASCII コードとみなして 1 文字表示する。<br>●入力　A … ASCII コード・データ<br>●出力　なし<br>●レジスタ　A′ 以外保存 |
| 142F | 文字列表示<br>●DE が示すアドレスからの文字列データを表示。<br>文字列の終わりは 00H<br>●入力　DE …文字列スタート・アドレス<br>●出力　なし<br>●レジスタ　A, A′ 以外保存 |
| 143C | TAB 実行<br>●次の TAB 位置にカーソルを移動する。<br>●レジスタ　A 以外保存 |
| 1446 | 改行実行<br>●改行を行う。<br>●レジスタ　A 以外保存 |
| 画面表示サブルーチン　2　（出力切り換えなし） | |
| 1321 | ファイル名表示<br>●「ファイル名（13文字）．拡張子（3文字）」で表示する。<br>●入力　HL …ファイル名格納アドレス－1<br>●出力　なし<br>●レジスタ　A, D 以外保存 |
| 1340 | 文字列表示<br>●HL が示すアドレスからの文字列データを表示。<br>●入力　HL …文字列スタート・アドレス<br>D …文字数<br>●出力　なし<br>●レジスタ　A, D 以外保存 |

| アドレス | 内　　容 |
|---|---|
| データ変換サブルーチン | |
| 1451 | 英小文字→英大文字変換<br>●A レジスタの値を ASCII コードとみなし，もしも英小文字なら英大文字に変換する。<br>●入力　A …変換したい ASCII コード<br>●出力　A …変換された ASCII コード<br>●レジスタ　A 以外保存 |
| 1143 | 1 文字数値変換<br>●DE で示されるメモリから 1 バイト(ただし先頭のスペースは無視する)を16進表記の文字とみなし，数値に変換する。<br>●入力　DE …文字列先頭アドレス<br>●出力　A …変換数値データ<br>DE … DE ＋ 1 ＋スペース個数<br>キャリーフラグ… 0 のとき正常<br>1 のときエラー（0 ～ 9，A ～ F 以外の文字があった）<br>●レジスタ　A, DE 以外保存 |
| 115E | 1 バイト数値取り出し<br>●DE が示す16進表記の文字列から 1 バイト取り出す。<br>ただし先頭に“；”が付いていたら次の文字の ASCII コードをデータとする。<br>●入力　DE …文字列先頭アドレス<br>●出力　A …データ<br>キャリーフラグ… 0 のとき正常（DE は次の文字をさす）<br>1 のときエラー（DE は保存） |
| 111F | 2 バイト数値取り出し<br>●DE が示す16進表記文字列から 2 バイト取り出す。<br>●入力　DE …文字列先頭アドレス<br>●出力　HL …データ<br>キャリーフラグ… 0 のとき正常<br>1 のときエラー（DE, HL 保存）<br>●レジスタ　HL, DE, A 以外保存 |

<table>
<tr><th>アドレス</th><th>内　　容</th></tr>
<tr><td>12A6</td><td>2バイト数値の3回取り出し<br>●S. T. コマンドでのスタートアドレス，バイト数，XX アドレスを取り出す。<br>●入力　DE …文字列先頭アドレス<br>●出力　DE …第1アドレス（スタートアドレス）<br>HL …第3アドレス<br>BC …第2アドレス−第1アドレス＋1（バイト数）<br>キャリーフラグ… 0のとき正常<br>1のときエラー<br>●レジスタ　HL, BC, DE, A 以外保存</td></tr>
<tr><td colspan="2">その他のサブルーチン</td></tr>
<tr><td>12D5</td><td>プリンタの改行実行<br>●プリンタへの改行を行う。<br>プリンタのエラーがあったときエラー処理ルーチンへジャンプしてしまう。<br>●入力　なし<br>●出力　なし<br>●レジスタ　A 以外保存</td></tr>
<tr><td>12DC</td><td>プリンタへの1文字出力<br>●プリンタへ1文字出力する。<br>●入力　A …出力する ASCII コード<br>●レジスタ　全レジスタ保存</td></tr>
<tr><td>1315</td><td>プリンタへの TAB 実行<br>●プリンタヘッドを次の TAB 位置へ移動させる。<br>●入力　なし<br>●出力　なし<br>●レジスタ　A 以外保存</td></tr>
<tr><td>134E</td><td>ファイル名比較<br>●1481H 以降にあるファイル名・パスワードと HL で示されるファイル名・パスワードを比較し，一致しているかどうか調べる。<br>●入力　HL …比較ファイル名格納アドレス−1<br>●出力　ゼロフラグ… 0のとき不一致<br>1のとき一致<br>●レジスタ　A 以外保存</td></tr>
</table>

| アドレス | 内　容 |
|---|---|
| 138B | ファイル名セット<br>●1480H 以降のインフォメーション・ワーク・エリアにファイル名，パスワード，日付をセットする。<br>●入力　DE …ファイル名格納アドレス<br>　　　　　“：”の次からがファイル名となる。<br>　　　　　“：”がみつかるまで DE をインクリメントする。<br>●出力　なし<br>●レジスタ　A, BC, DE, HL 以外保存<br>注：“：”のサーチが不用なときは DE レジスタに（ファイル名格納アドレス－1）をセットし 1349H をコールする。 |

表2-16 **各種エントリー**

| アドレス | 内　容 |
|---|---|
| 0FE2 | BASIC の MON コマンドエントリー，エラージャンプ先をモニタ内に変更し，BASIC での1行入力文字数，スタック・ポインタを退避，出力デバイスを CRT にセットしてモニタにジャンプする。 |
| 1000 | モニタ・スタート |
| 102D | モニタ・エラー処理<br>「ERR？」と表示しテープ・ストップして，1000H にジャンプする。 |
| 1061 | P コマンド<br>1472H（出力デバイス・フラグ）を反転する。 |
| 106A | S コマンド |
| 109A | L コマンド |
| 10E1 | E コマンド |
| 10F0 | R コマンド<br>1行入力文字数，エラージャンプ先を戻し，スタック・ポインタをFDF6Hにセットしてリターン。よってリターン・アドレスは FDF6, FDF7H にある。 |
| 1186 | G コマンド |
| 118B | D コマンド |
| 121D | M コマンド |
| 1253 | F コマンド |
| 12BF | T コマンド |

**表2 17 ワーク・エリア**

| アドレス | 内　　容 |
|---|---|
| 1043～1060 | コマンド・ワーク・エリア |
| 145A～1461 | メッセージ " Writing " |
| 1462～1469 | メッセージ " Found ␣␣" |
| 146A～1471 | メッセージ "Skip ␣␣␣" |
| 1472 | 出力ディバイスフラグ<br>00H → CRT<br>01H → プリンタ |
| 1473～1477 | メッセージ " ERR ? " |
| 147E～147F | エラージァンプ先 |
| 1480～149F | インフォメーション・バッファ |
| FF00～ | キー入力バッファ |

# 2-10 キー入力

X1は，先行入力やON KEY GOSUBによるキー入力割り込みなどの機能を持っています。本章の最後として，このキー入力処理についてみていきましょう。

## 2-10-1 行連続フラグ

IOCSには1行入力用サブルーチンとしてBASIC用のBINPUT(015AH)と通常のINPUTF(0003H)が用意されています。**リスト2-11**はBINPUTの逆アセンブル・リストです。これを見るとキー入力の前に次行との連続を切り，CONSOLE命令の指定座標とのチェックを行っていることがわかります。さらに入力スタート時のX座標以降しか取り組んでいません。(つまり，プロンプトは取り込まれない)。

この解析から，現在カーソルのある行が次行とつながっているのかを調べるワーク・エリアは

| ページ0のとき 00A9H～00C2H |
|---|
| ページ1のとき 00C3H～00DCH |

にあることがわかります。X1は25行表示できるので，1行に1バイトを割り当て，ここに00Hが入っていれば次行とはつながっていない，01Hが入っていればつながっているということを表しています。

リスト2 11

```
Addr   Code      Mnemonic

015A   E5        PUSH    HL
015B   2A0E00    LD      HL,(000EH)   ; H ←カーソル Y 座標, L ← X 座標
015E   E5        PUSH    HL
015F   D5        PUSH    DE
0160   CD2105    CALL    0521H        ; HL に行連続テーブルが入る
0163   D1        POP     DE
0164   3600      LD      (HL),00H     ; 次の行との連続をなくす
0166   E1        POP     HL
0167   CD7C01    CALL    017CH        ; 1 行キー入力
016A   380E      JR      C,017AH      ; 中断したとき
016C   3A1E00    LD      A,(001EH)    ; A ← CONSOLE の開始 X 座標
016F   95        SUB     L
0170   3008      JR      NC,017AH     ;
0172   ED44      NEG                  ;  CONSOLE およびプロンプト
0174   6F        LD      L,A          ;  分だけ DE の値を増加する
0175   2600      LD      H,00H        ;
0177   19        ADD     HL,DE        ;
0178   EB        EX      DE,HL        ;
0179   B7        OR      A
017A   E1        POP     HL
017B   C9        RET
```

## 2-10-2 割り込みとキー入力バッファ

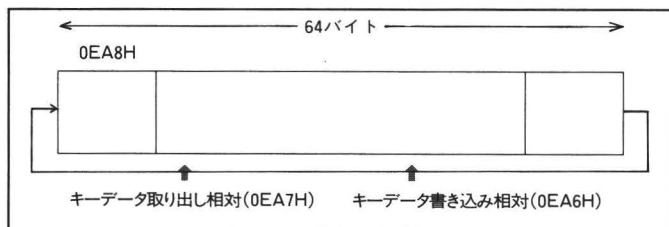
HuBASIC では通常割り込みによってキー入力を処理しています。このため先行入力ができ，CPU が他の処理を行っている間でもキーの読み込みが優先して行われています。このキーデータはいったんキーバッファに入りキーデータが必要になったときはこのバッファから取り出しています。

キーバッファは 0EA8H から64文字分あります。先行キー入力が63文字まで可能なのはそのためです（最後の１バイトはエンド・コード 00H)。

キーデータの書き込み，取り出しは 0EA8H をゼロとする相対アドレスで示され，それぞれ 0EA6H，0EA7H に入っています（図2-23)。

0EA5H が，キーバッファクリア・フラグになっていて，この値が 0 以外のときはキーバッファ用ポインタ（0EA6H と 0EA7H)をクリアした後にキー入力が行われます。これは

BINPUT でも INPUTF でも有効です。KBUF ON/OFF 命令はこの値を操作しています（KBUF 命令は BASIC マニュアルには載っていませんが，その名の通りキーバッファを使うか否かの選択を行う命令です)。KBUF ON のときは 00H に OFF のときは 01H に設定されます。

図2 23 **キーバッファ（リング・バッファ構造）**

# 2-10-3 ファンクション・キー

ファンクション・キーに登録されるデータは 0F42H ～0FE1H にあり，初期は**図2-24**のようになっています(テープ・バージョン)。

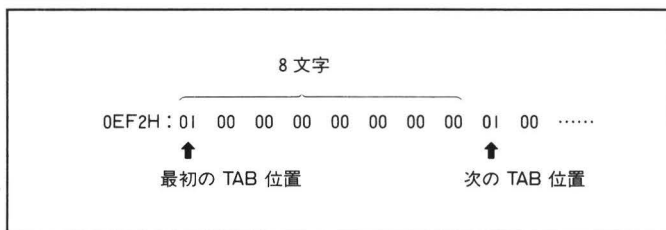
図2 24 **ファンクション・キーデータ**

```
:0F42=05 41 55 54 4F 0D 00 00 /.AUTO...
:0F4A=00 00 00 00 00 00 00 00 /........
:0F52=07 3F 54 49 4D 45 24 0D /.?TIME$.
:0F5A=00 00 00 00 00 00 00 00 /........
:0F62=03 4B 45 59 00 00 00 00 /.KEY....
:0F6A=00 00 00 00 00 00 00 00 /........
:0F72=06 4C 49 53 54 1A 0D 00 /.LIST...
:0F7A=00 00 00 00 00 00 00 00 /........
:0F82=06 52 55 4E 20 20 0D 00 /.RUN   ..
:0F8A=00 00 00 00 00 00 00 00 /........
:0F92=06 4C 4F 41 44 20 0D 00 /.LOAD ..
:0F9A=00 00 00 00 00 00 00 00 /........
:0FA2=06 57 49 44 54 48 20 00 /.WIDTH .
:0FAA=00 00 00 00 00 00 00 00 /........
:0FB2=05 43 48 52 24 28 00 00 /.CHR$(..
:0FBA=00 00 00 00 00 00 00 00 /........
:0FC2=06 50 41 4C 45 54 20 00 /.PALET .
:0FCA=00 00 00 00 00 00 00 00 /........
:0FD2=05 43 4F 4E 54 0D 00 00 /.CONT...
:0FDA=00 00 00 00 00 00 E2 0F /......ヰ.
```

ファンクション・キーを割り込み処理として使用する場合(ON KEY GOSUB 命令), F1～F10 の飛び先番号は, 361EH (3650H) ～3631H (3663H) に登録されます。格納データは16進 2 バイトの行番号値です。ON KEY GOSUB 命令を実行した時点で設定され, 行番号の指定がないファンクション・キーのところには 0000H が格納されます。

# 2-10-4 TAB キー

X1 では TAB キーの間隔は 8 文字に設定されています。TAB 位置検出用のデータ・エリアは 0EF2H～0F41H にあります。このエリアは画面の X 軸方向の対応していて, その内容の 01H のところが TAB 位置になります(**図2-25**)。そのため, ここを書き換えることで自由に TAB 位置を設定できます。

図2-25 **TAB 位置設定用ワーク・エリア**

# 第3章

# 画面構成



X1の大きな特徴のひとつとして，豊富な画面表示機能があげられます。本章では，この機能の中心となる**CRT**コントローラの解説を始め，スーパインポーズ，プライオリティなど**X1**特有の機能について述べていきます。また，**PCG**の定義，漢字の表示など有用なサブルーチンも数多く紹介します。

# 3-1 CRT コントローラ

X 1 ではCRT コントローラとして,もっともメジャーなHD46505SP-2 を採用しています。

このCRTC は,本来68系の CPU 用につくられた LSI ですが80系でも良く使用されています。画面に対するメモリのアドレスを出力するのが主な機能ですが,その他にも**図3-1** のような機能を持っています。

図3-1 **HD46505 SP-2 の機能**

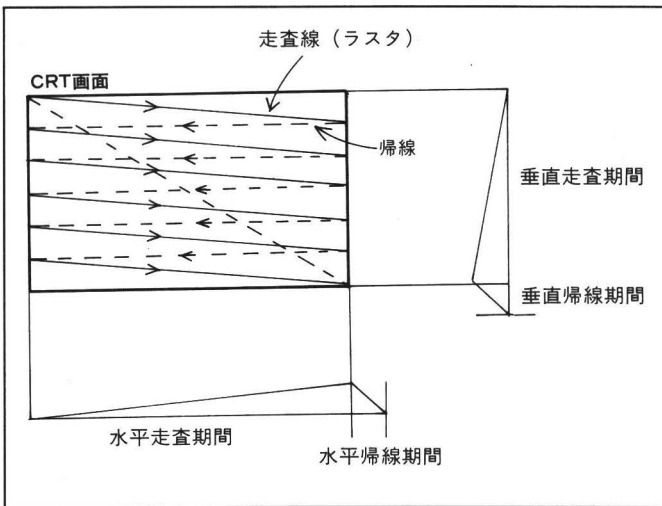
| | |
|---|---|
| 画面構成 | 水平走査の周期を1文字単位で設定可<br>垂直走査の周期を1行時間単位で設定可<br>垂直走査の周期をラスタ単位で微調整可<br>1行の表示文字数設定可<br>1画面の表示行数設定可<br>1行のラスタ数設定可<br>水平方向表示開始位置設定可<br>垂直方向表示開始位置設定可<br>水平同期信号パルス幅1文字単位で設定可<br>垂直同期信号パルス幅1ラスタ単位で設定可 |
| カーソル表示 | カーソル表示位置設定可<br>カーソルの形状設定可<br>カーソルのブリンク周期(16, 32フィールド)選択可 |
| スキャンモード | ノンインターレス・モード<br>インターレス・シンク・モード<br>インターレス・シンク&ビデオ・モード |
| ライトペン | ライトペン・レジスタ内蔵 |
| アドレッシング | リフレッシュ・メモリ・アドレス出力 |
| スタートアドレス | スタートアドレス・レジスタ内蔵によりページング,スクローリング可能 |

これらの機能のうちX1では，

- 水平・垂直同期信号，表示タイミング
- 表示画面の大きさ設定
- VRAMのリフレッシュ

などに使っています。

ラスタ・スキャン方式としてはノン・インターレースモードを使っています（**図3-2**）。ちなみに，インターレースモードというのは，走査線を偶数フィールドと奇数フィールドの2回に分けてコマ数を2倍にし，チラツキを少なくするモードです。

**図3-2 ノン・インターレースモード**



“垂直走査期間＋垂直帰線期間”が垂直同期信号幅に相当し，X1では約188 μs，また水平同期信号幅は約4.47 μsです。

CRTCは内部にAR（アドレス・レジスタ）を除いて計18個のレジスタを持っており，これらに値を設定することで各種機能を行わせます。ARはこれらのレジスタを指定するために使います。

## 3-1-1 CRTCのレジスタ

CRTCの18個のレジスタは次のとおりです。

**●R0　水平総文字数レジスタ**

水平走査の周期をセットします。設定値は,“表示されない部分も含めた，水平方向の１掃引時間に対する文字数”から１を引いた値です。

**●R1　水平表示文字数レジスタ**

画面に表示される１行あたりの文字数をセットします。キャラクタ・カウンタがこの値を越えると,表示が禁止されます。

**●R2　水平同期位置レジスタ**

水平同期信号の出力位置をセットします。

設定値は,“水平同期位置の文字数”－１です。

**●R3　同期パルス幅レジスタ**

下位４ビットで水平同期信号のパルス幅（水平１文字時間)を，上位４ビットで垂直同期信号のパルス幅（水平１走査時間）をセットします。

**●R4　垂直総文字数レジスタ**

垂直走査の周期をセットします。設定値は,“垂直方向の１掃引時間に対する文字数”－１です。ライン・カウンタの値がこの値と一致すると，次のコマへ移ります。

**●R5　総ラスタ調整**（トータル・ラスタ・アジャスト）

１フレームあたりの総ラスタ数を微調整するための付加ラスタ数をセットします。

１フレームの掃引周期はR４によって自動的に決まりますが，微調整のためにさらにこのレジスタの本数分だけ加えられます。

**●R6　垂直表示文字数レジスタ**

画面上に表示する行数をセットします。ライン・カウンタがこの値と一致すると画面への表示を禁止します。

**●R7　垂直同期位置レジスタ**

垂直同期信号の出力位置をセットします。ライン・カウンタがこの値と一致したときに垂直同期パルスを出力します。

### ●R8　インターレース & スキューレジスタ

画面のインターレース設定とDISPTMG信号・CUDISP 信号のスキュー（遅延）の設定を行います。

設定値については**図3-3**を参考にしてください。

**図3-3 インターレース，スキューレジスタ**

インターレース・モードの選択

| V | S | モード |
|---|---|---|
| 0 | 0 | ノンインターレース・モード |
| 1 | 0 | ノンインターレース・モード |
| 0 | 1 | インターレース・シンク・モード |
| 1 | 1 | インターレース・シンク & ビデオ・モード |

ビット

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| $C_1$ | $C_0$ | $D_1$ | $D_0$ | × | × | V | S |

| $D_1$ | $D_0$ | DISPTMG信号 |
|---|---|---|
| 0 | 0 | スキューなし |
| 0 | 1 | 1文字スキュー |
| 1 | 0 | 2文字スキュー |
| 1 | 1 | 出力されない |

| $C_1$ | $C_0$ | CUDISP信号 |
|---|---|---|
| 0 | 0 | スキューなし |
| 0 | 1 | 1文字スキュー |
| 1 | 0 | 2文字スキュー |
| 1 | 1 | 出力されない |

### ●R9　最大ラスタアドレス・レジスタ

スペースを含めた1行のラスタ数をセットします。設定値は“1行のラスタ数”－1です。

### ●R10　カーソル・スタート・ラスタレジスタ

上位2ビットでカーソルの表示モードを，下位5ビットでカーソル上端のラスタアドレスを指定します。

X1 ではこの機能は使用していません。

カーソル表示モードの設定値は**図3-4**を参考にしてください。

図3-4 カーソルの表示モード

| $2^6$ | $2^5$ | カーソル表示 |
|---|---|---|
| 0 | 0 | カーソルはブリンクしない |
| 0 | 1 | カーソルは表示されない |
| 1 | 0 | 16フレーム間隔でブリンクする |
| 1 | 1 | 32フレーム間隔でブリンクする |

**●R11　カーソル・エンド・ラスタレジスタ**

カーソルの下端のラスタアドレスをセットします。R 9，10とは次のような関係があります。

R10(カーソル・スタート)≦R11(カーソル・エンド)≦R9(最大ラスタ)

**●R12，R13　スタート・アドレスレジスタ**

リフレッシュ・メモリの読み出し先頭アドレスをセットします。このアドレスから表示も開始されます。

**●R14，R15　カーソル・アドレスレジスタ**

カーソルの表示アドレスをセットします。上位2ビットは無意味です。X1 では使用していません。

**●R16，R17　ライトペン・レジスタ**

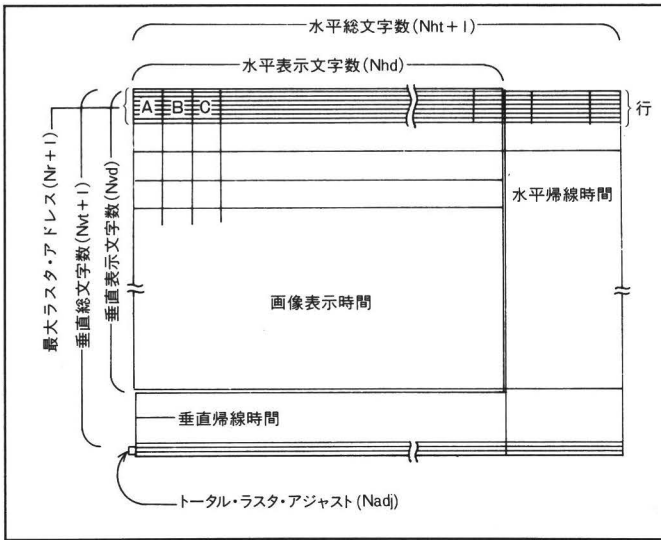
ライトペンの検出アドレスを記憶します。LPSTB がアクティブになった時点の画面のアドレスが入ります。

以上レジスタ内容とその設定値を**図3-5**にまとめておきます。**図3-6**の CRT 画面構成と合わせて見てください。

図3-5 CRTコントローラのレジスタ

| レジスタ | レジスタ名 | 書き込み値 | レジスタ | レジスタ名 | 書き込み値 |
|---|---|---|---|---|---|
| R0 | 水平総文字数 | Nht | R9 | 最大ラスタ・アドレス | Nr |
| R1 | 水平表示文字数 | Nhd | R10 | カーソル・スタート・ラスタ | |
| R2 | 水平同期位置 | Nhsp | R11 | カーソル・エンド・ラスタ | |
| R3 | 水平同期パルス幅 | Nhsw | R12 | スタート・アドレス(H) | |
| R4 | 垂直総文字数 | Nvt | R13 | スタート・アドレス(L) | |
| R5 | トータル・ラスタ・アジャスト | Nadj | R14 | カーソル(H) | |
| R6 | 垂直表示文字数 | Nvd | R15 | カーソル(H) | |
| R7 | 垂直同期位置 | Nvsp | R16 | ライト・ペン(H) | |
| R8 | インターレース・モード | | R17 | ライト・ペン(L) | |

(注) Nhd＜Nht　Nvd＜Nvt

図3 6 **CRTの画面構成**

# 3-1-2 画面モードの設定

CRTCのアドレス・レジスタARはI/Oアドレスの1800Hに，レジスタ・データは1801Hに割り当てられています。あるレジスタに値を書き込むときは，次のような手順が必要です。

```
LD    BC, 1800H    ;アドレス・レジスタのI/Oアドレス
LD    A, [レジスタ番号] ;書き込むレジスタを指定する。
OUT   (C), A
LD    BC, 1801H    ;データ用I/Oアドレス
LD    A, [データ]    ;書き込むデータ
OUT   (C), A
```

X1は起動後，かならずCRTCの初期設定を行わなければなりません。18個のレジスタに適切な値を書き込めば良いわけです。

画面モードの変換（たとえば40文字モードを80文字モードに変えるなど）では，変換時に値が異なるレジスタだけを変

えれば良いでしょう。しかし，通常は全レジスタ・データ(18バイト）をテーブルに持たせておき，テーブル上の値を変更させてから，全レジスタに書き込み直すという方法が行われます。このほうが結局はプログラムが短かくなります。

**リスト3-1**が実際に40字モードを設定するためのプログラムです。I/O アドレス 1A02H は8255②のポート C に割り当てられており，その内容は**図3-7**のようになっています。

リスト3 1　No. 1

```
 1:                   ;
 2:                   ;      40 ﾓｼﾞ ﾓｰﾄﾞ
 3:                   ;                              LIST 3-1
 4:   0003 =          INPUTF  EQU     0003H
 5:                   ;
 6:   C000                    ORG     0C000H
 7:                   ;
 8:   C000 1600               LD      D,0            ;Register NO. Top
 9:   C002 1E12               LD      E,18           ;Counter
10:   C004 2123C0             LD      HL,DATA        ;Data top
11:   C007 010018     LOOP:   LD      BC,1800H       ;Register NO. I/O Address
12:   C00A ED51               OUT     (C),D
13:   C00C 03                 INC     BC
14:   C00D 7E                 LD      A,(HL)
15:   C00E ED79               OUT     (C),A          ;Data Set
16:   C010 23                 INC     HL
17:   C011 14                 INC     D
18:   C012 1D                 DEC     E
19:   C013 20F2               JR      NZ,LOOP
20:   C015 01031A             LD      BC,1A03H
21:   C018 3E0D               LD      A,0DH          ;Clock Change
22:   C01A ED79               OUT     (C),A
23:                   ;
24:   C01C 1100C2             LD      DE,0C200H      ;Time Wait
25:   C01F CD0300             CALL    INPUTF
26:                   ;
27:   C022 C9                 RET
28:                   ;
29:   C023            DATA:
30:   C023 37                 DEFB    37H            ;R0  H Length Max
31:   C024 28                 DEFB    28H            ;R1  H Display Length
32:   C025 2D                 DEFB    2DH            ;R2  HSYNC Position
33:   C026 34                 DEFB    34H            ;R3  SYNC Pulse
34:   C027 1F                 DEFB    1FH            ;R4  V Length Max
35:   C028 02                 DEFB    02H            ;R5  Luster Adjust
36:   C029 19                 DEFB    19H            ;R6  V Display Length
37:   C02A 1C                 DEFB    1CH            ;R7  VSYNC Position
38:   C02B 00                 DEFB    00H            ;R8  Interlace
39:   C02C 07                 DEFB    07H            ;R9  Luster Max
40:   C02D 00                 DEFB    00H            ;R10 Cursor Start
41:   C02E 00                 DEFB    00H            ;R11 Cursor End
42:   C02F 00                 DEFB    00H            ;R12 Start Address H
43:   C030 00                 DEFB    00H            ;R13 Start Address L
44:   C031 00                 DEFB    00H            ;R14 Cursor H
45:   C032 00                 DEFB    00H            ;R15 Cursor L
```

No. 2

```
46:    C033 00                DEFB    00H          ;R16 Light Pen H
47:    C034 00                DEFB    00H          ;R17 Light Pen L
48:               ;
49:    C035                   END
```

図3-7 I/O アドレス 1A02H

8255② ポートC（出力）1A02H

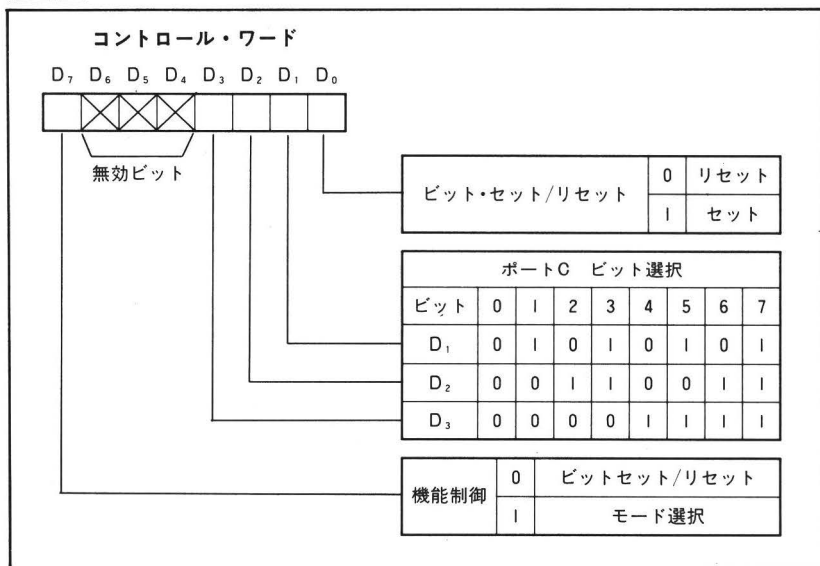
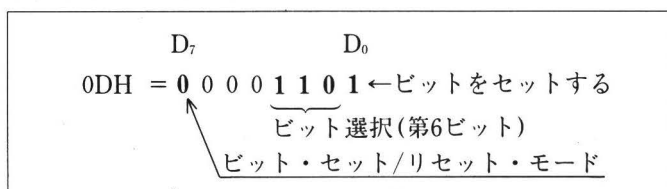
| | | |
|---|---|---|
| $PC_7$ | プリンタ用 $\overline{\text{STROBE}}$ | アクティブL |
| $PC_6$ | 80/40字モード基本クロック切り換え | H:40字　L:80字 |
| $PC_5$ | I/O アクセス・モード切り換え | ↓ |
| $PC_4$ | スムーズ・スクロール信号 | アクティブL |
| $PC_3$ | | |
| $PC_2$ | | |
| $PC_1$ | | |
| $PC_0$ | カセットテープ書き込み信号 | |

ビット6が80字/40字の基本クロック切り換えになっているので，F0Hを出力して40字モードに切り換えます。

```
              PC6
F0H  = 1 1 1 1 0 0 0 0
         ↑
         └40字モード・クロック設定
```

また，Cポートが出力の場合、そのビットの位置を指定してセットまたはリセットすることが8255では可能なので，その機能を使用して40字モード・クロックに切り換えてもかまいません。X1で使われているプログラムのほとんどが後者の方法をとっているので，ここでもそれにしたがいます。

ポートCをビット単位でセット，リセットさせるには，**図3-8**のような内容のコントロール・ワードをコントロール・レジスタ設定ポートに送ることで実現できます。

図3-8 ポートCのコントロール・ワード

ここでは，1A03Hがコントロール・ポートにあたります。このI/Oアドレスに0DHを送ることで40字モード・クロックに切り換わります。

ビット制御のときは$D_7$はつねに0でなければなりません。ちなみに，80字モード・クロックにする場合のコントロール・ワードは0CHです。

80字/40字におけるCRTCのレジスタ設定値は**図3-9**のようになっています。

X1の回路図を見るとわかると思いますが，LPSTB端子(ライトペン端子)はGNDに接続され，CUDSP端子(カーソル表示端子)は開放されているため，ライトペン・レジスタとカーソル関係のレジスタはどんな値を書き込んでも無意

味です。

他のレジスタは値をいろいろと変えて試してみるとおもしろいでしょう。

図3 9 **CRTCのレジスタ設定値**

| レジスタ信号 | 40文字モード(16進) | 80文字モード(16進) | 内容 |
|---|---|---|---|
| 0 | 37 | 6F | 水平総文字数 |
| 1 | 28 | 50 | 水平表示文字数 |
| 2 | 2D | 59 | 水平同期位置 |
| 3 | 34 | 38 | 同期パルス幅 |
| 4 | 1F | 1F | 垂直総文字数 |
| 5 | 02 | 02 | トータル・ラスタ・アジャスト |
| 6 | 19 | 19 | 垂直表示文字数 |
| 7 | 1C | 1C | 垂直同期位置 |
| 8 | 00 | 00 | インタレース |
| 9 | 07 | 07 | 最大ラスタ・アドレス |
| 10 | 60 | 60 | カーソル・スタート・ラスタ |
| 11 | 07 | 07 | カーソル・エンド・ラスタ |
| 12 | 00 | 00 | スタート・アドレス(H) |
| 13 | 00 | 00 | スタート・アドレス(L) |
| 14 | 00 | 00 | カーソル(H) |
| 15 | 00 | 00 | カーソル(L) |
| 16 | 00 | 00 | ライトペン(H) |
| 17 | 00 | 00 | ライトペン(L) |

# 3-1-3 スムーズ・スクロール

CRTC制御の一例としてスムーズ・スクロールを行ってみましょう。

スムーズ・スクロールとは HuBASIC の SCROLL 命令と同様で，スーパーインポーズを行ったときにコンピュータ画面だけを上下に間断なくスクロールさせる機能です。

スクロールを実行させるには8225②のポート $C_4$ と CRTC のレジスタ5（総ラスタ調整）を使って次のように行います。8255②のポートCについては**図3-7**を参考にしてください。

**1** CRTCのレジスタ5に値を書き込みます（**図3-10**）。これで上下方向のスクロール速度が決まります。

**図3-10 スクロール速度**

| レジスタ5の値(10進) | 0←2←4←6 | 8→10→12→------→30 |
|---|---|---|
| スクロール方向 | 上　方　向 | 下　方　向 |
| スクロール速度 | 速い← | →速い |

レジスタ5で指定されたラスタ本数だけ余分に垂直方向掃引に時間がかかるためスクロールしているように表示されます。なお，レジスタ5の設定値は必ず偶数でなければなりません。

**2** 8255②のポート $C_4$ を“L”に落とし，スクロールをONにします。

スクロールを停止させるには次のように逆の手順で行います。

**1** CRTCのレジスタ5に初期値02Hを書き込みます。

**2** 8255②ポート $C_4$ を“H”にし，スクロールをOFFにします。

**リスト3-2**がスクロール開始のプログラムです。Aレジスタには0～15の値を入れます。2倍することにより必ず偶数が入るようにしてあります。

スムーズ・スクロールはテレビ画面とコンピュータ画面を重ね合わせたときのみ有効です。

**リスト3-3**はスクロールを停止させるプログラムです。

CRTCのレジスタ5(総ラスタ調整)の値を順次大きくしていくことによって，ほんのわずかですがスーパインポーズ状態でなくともスムーズにスクロールさせることが可能です。

リスト3 2

```
 1:                    ;
 2:                    ;         Smooth Scroll
 3:                    ;                                  LIST 3-2
 4:                    ;
 5:    C000                      ORG     0C000H
 6:                    ;
 7:    C000 3E02                 LD      A,2              ;Scroll Speed 0-15
 8:    C002 E60F                 AND     0FH
 9:    C004 87                   ADD     A,A
10:                    ;
11:    C005 010018               LD      BC,1800H         ;CRTC R5 Set
12:    C008 1605                 LD      D,05H
13:    C00A ED51                 OUT     (C),D
14:    C00C 03                   INC     BC
15:    C00D ED79                 OUT     (C),A
16:                    ;
17:    C00F 01031A               LD      BC,1A03H
18:    C012 3E08                 LD      A,08H
19:    C014 ED79                 OUT     (C),A            ;Scroll ON
20:                    ;
21:    C016 C9                   RET
22:                    ;
23:    C017                      END
```

リスト3 3

```
 1:                    ;
 2:                    ;         Scroll Stop
 3:                    ;                                  LIST 3-3
 4:                    ;
 5:    C000                      ORG     0C000H
 6:                    ;
 7:    C000 010018               LD      BC,1800H
 8:    C003 110205               LD      DE,0502H
 9:    C006 ED51                 OUT     (C),D
10:    C008 03                   INC     BC
11:    C009 ED59                 OUT     (C),E
12:                    ;
13:    C00B 01031A               LD      BC,1A03H
14:    C00E 3E09                 LD      A,09H
15:    C010 ED79                 OUT     (C),A
16:                    ;
17:    C012 C9                   RET
18:                    ;
19:    C013                      END
```

**リスト3-4**がそのプログラムで，レジスタ５の値は20Hが限度です。さらに連続してスクロールさせたいときはレジスタ７（垂直同期位置）をうまく変化させると良いでしょう。

また，スクロールのスピードは調節できませんが，レジスタ９（最大ラスタアドレス）を0FHにすることで画面がスクロールしているように見せることができます。

リスト3 4

```
 1:                  ;
 2:                  ;       Smooth Scroll non Superimpose
 3:                  ;                                    LIST 3-4
 4:                  ;
 5:   C000                   ORG     0C000H
 6:                  ;
 7:   C000 2E05              LD      L,05H
 8:   C002 3E00              LD      A,00H
 9:   C004 010018   LOOP1:   LD      BC,1800H
10:   C007 ED69              OUT     (C),L
11:   C009 03                INC     BC
12:   C00A ED79              OUT     (C),A
13:   C00C 3C                INC     A
14:                  ;
15:   C00D 012000            LD      BC,0020H
16:   C010 05       LP:      DEC     B
17:   C011 20FD              JR      NZ,LP
18:   C013 0D                DEC     C
19:   C014 20FA              JR      NZ,LP
20:                  ;
21:   C016 FE20              CP      20H
22:   C018 20EA              JR      NZ,LOOP1
23:   C01A C9                RET
24:                  ;
25:   C01B                   END
```

# 3-2 テキスト画面とアトリビュート

テキスト画面とは，通常の文字やプログラマブル・キャラクタ・ジェネレータ（PCG）に定義された文字を表示する画面です。グラフィックに対して独立したVRAMを使用しているのでグラフィック画面との混在が可能です。

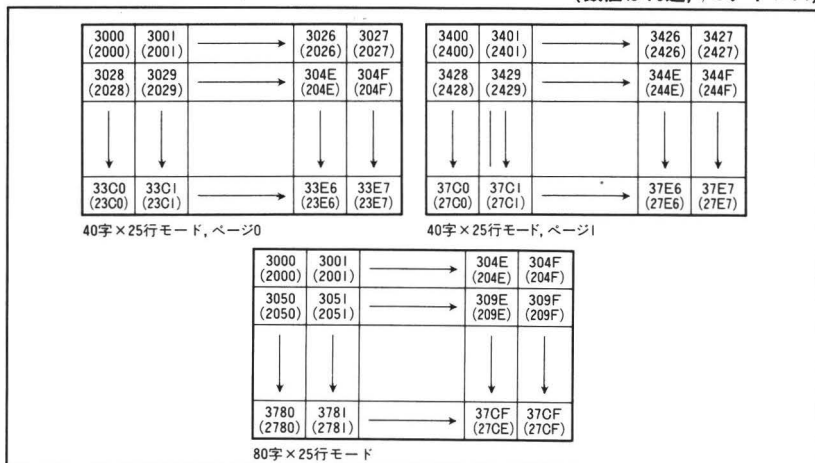
テキスト画面では，次の2つのモードが選択できます。

❶80字×25行　1画面（ページ0）

❷40字×25行　2画面（ページ0，1）

また，テキストVRAMと同じ大きさのアトリビュート(属性) VRAMがあり，色の指定，キャラクタ・ジェネレータROMかPCG RAMかの切り換え，文字サイズ，文字反転，文字点滅を1文字毎に指定できます。

テキスト画面の表示位置は，テキスト・アトリビュートVRAMのアドレスと1対1に対応しており，**図3-11**のように割り当てられています。カッコ内の数値がアトリビュートVRAMのアドレスです。

**図3-11 テキスト画面の表示位置とアトリビュート**　（数値は16進，I/Oアドレス）

テキスト・アドレスとアトリビュート・アドレスの関係は次式のようになっています。

| テキスト・アドレス＝アトリビュート・アドレス＋1000H |
|---|

実際のアドレス変換方法としては，BC ペア・レジスタにテキスト・アドレスが入っている場合，

```
LD    BC, [テキスト・アドレス]
RES   4,  B
```

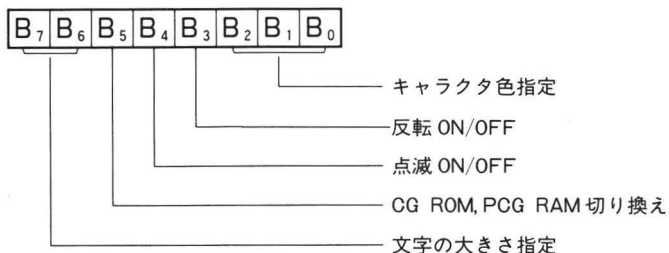
となります。このプログラムを実行することで，BCレジスタにはテキスト・アドレスに対応したアトリビュート・アドレスを得ることができます。

アトリビュートからテキスト・アドレスに変換する場合も同様に，

```
LD    BC, [アトリビュート・アドレス]
SET   4,  B
```

で求められます。

アトリビュート VRAM の各ビットは**図3-12**のようになっています。

**図3-12 アトリビュートのビット内容**

また，その設定値は**図3-13**のような意味を持ちます。

テキスト画面のアクセス例を**リスト3-5**に示します。これは，Hレジスタに X 座標値，L レジスタに Y 座標値，A レジスタに ASCII コードを入れ，D レジスタにアトリビュート・コードを設定してコールします。

この例では画面中央に，文字“X”をキャラクタ色赤で点滅させながら表示させています。

ビット 3 で反転を指定した場合はキャラクタ色とバック色がいれかわります。バック色の初期値は黒（透明）で，パレット機能で切り換えることができます。

**図3-13 アトリビュートのビット内容と設定値**

**●キャラクタ色**

| $B_2$ | $B_1$ | $B_0$ | 設定色 |
|---|---|---|---|
| 0 | 0 | 0 | 黒 |
| 0 | 0 | 1 | 青 |
| 0 | 1 | 0 | 赤 |
| 0 | 1 | 1 | マゼンタ |
| 1 | 0 | 0 | 緑 |
| 1 | 0 | 1 | シアン |
| 1 | 1 | 0 | 黄 |
| 1 | 1 | 1 | 白 |

**●反転**

| $B_3$ | 機　能 |
|---|---|
| 0 | $B_0$～$B_2$で指定したキャラクタカラーで表示 |
| 1 | $B_0$～$B_2$で指定したキャラクタカラーの補色で表示します。 |

**●点滅**

| $B_4$ | 機　能 |
|---|---|
| 0 | 点滅せずに表示 |
| 1 | キャラクタが0.5秒周期で点滅 |

**●表示モード**

| $B_5$ | 設定モード |
|---|---|
| 0 | 標準文字モード (CG ROM) |
| 1 | ユーザー定義文字モード (PCG RAM) |

**●キャラクタの大きさ**

| $B_7$ | $B_6$ | 機　能 |
|---|---|---|
| 0 | 0 | ノーマル文字 |
| 0 | 1 | 垂直 2 倍文字 |
| 1 | 0 | 水平 2 倍文字 |
| 1 | 1 | 垂直水平 2 倍文字 |

リスト3 5

```
 1:                      ;
 2:                      ;         Text & Attribute for 80ｼﾞ ﾓｰﾄﾞ
 3:                      ;                                  LIST 3-5
 4:                      ;
 5:                      ;                 Out status    BC = Text VRAM location
 6:                      ;                 Store         HL,DE
 7:                      ;
 8:    C000                        ORG     0C000H
 9:                      ;
10:    C000 2619                   LD      H,25             ;X Location
11:    C002 2E0A                   LD      L,10             ;Y Location
12:    C004 1658                   LD      D,58H            ;ASCII Code 'X'
13:    C006 1E12                   LD      E,12H            ;Attribute Data
14:    C008 CD0CC0                 CALL    PRINT
15:    C00B C9                     RET
16:                      ;
17:    C00C E5           PRINT:    PUSH    HL
18:    C00D D5                     PUSH    DE
19:    C00E E5                     PUSH    HL
20:    C00F 2600                   LD      H,0
21:                                                         ;----- HL * 80
22:    C011 29                     ADD     HL,HL            ;HL * 8
23:    C012 29                     ADD     HL,HL
24:    C013 29                     ADD     HL,HL
25:    C014 29                     ADD     HL,HL            ;HL * 10
26:    C015 44                     LD      B,H
27:    C016 4D                     LD      C,L
28:    C017 29                     ADD     HL,HL
29:    C018 29                     ADD     HL,HL
30:    C019 09                     ADD     HL,BC
31:                      ;
32:    C01A 010030                 LD      BC,3000H         ;Text Top Address
33:    C01D 09                     ADD     HL,BC
34:    C01E C1                     POP     BC
35:    C01F 48                     LD      C,B
36:    C020 0600                   LD      B,0
37:    C022 09                     ADD     HL,BC
38:    C023 44                     LD      B,H
39:    C024 4D                     LD      C,L
40:                      ;
41:    C025 D1                     POP     DE
42:    C026 ED51                   OUT     (C),D            ;Print ASCII
43:    C028 CBA0                   RES     4,B
44:    C02A ED59                   OUT     (C),E            ;Attribute Set
45:    C02C CBE0                   SET     4,B
46:    C02E E1                     POP     HL
47:    C02F C9                     RET
48:                      ;
49:    C030                        END
```

# 3-2-1 PCG VRAM

ユーザー定義可能なキャラクタ・ジェネレータ用のRAMは6Kバイト用意されており，これをPCG VRAMといいます。PCG VRAMはBlue, Red, Green各色に2Kバイトづつ割り当てられていて，ドット単位に8色指定できます。1キャラクタ分のドットパターンは8×8ドットであり，これが合計で256キャラクタ記憶できます。ですから，たとえばCG ROMの全キャラクタをPCGでおきかえることができます。

PCG VRAMはメイン・メモリ空間にもI/O空間にも直接割り当てられてはおらず，アドレス空間を占有しない分，特殊なアクセス方法を行わなければなりません。

PCG VRAMのアクセス方法を説明する前に，テキストVRAMの空エリアについて説明します。

画面に表示できる文字数は40字モードでは，40×25行＝1000文字です。ところが，40字モードの1画面に当てられているVRAMは1Kバイト＝1024バイトなのです。つまり，最後の24バイトは表示されることなく，あまっているのです。

また，80字モードでも80字×25行＝2000文字のところに2Kバイト＝2048バイトが当てられており，48バイトは未使用です。

しかし，CRTCのリフレッシュ・アドレス・メモリは11ビット構成であるため，2048個の表示アドレスを出力してきます。よって，画面には2000文字しか表示しませんが，残りの48バイトも垂直帰線期間中にアクセスされていて，そのデータがバス上に出力されているのです。

PCGの定義や読み出しは，このリフレッシュ・メモリ・アドレスの特性と余分なVRAMを利用して行っています。

具体的なPCG VRAMのアドレスは次のように決定されています。余分なVRAMに入っている8ビット・データ(ASCIIコード)がPCG VRAMの上位アドレスとなり，CRTCのラスタ・アドレスの下位8ビットがPCG VRAMの下位アド

レスとなります。

つまり，PCG データを送っている間にアクセスされるすべてのテキスト VRAM には，あらかじめ ASCII コードをセットしておかなければならないのです。

表示されない VRAM アドレスは40字モード，80字モードおよびページ数によって異なります。**図3-14** に表示されないテキスト VRAM アドレスとそのバイト数を示しておきます。また，**図3-15**には40字モード，ページ0における画面の使用状態を示します。

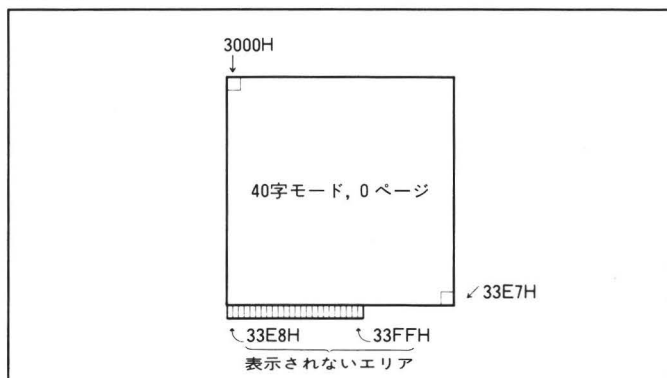
PCG VRAM はBlue, Red, Green の3種類あるわけで，それらの切りかえを行う必要があります。

各 PCG VRAM は**図3-16**のような I/O ポートにマッピングされていて，そのポートをアクセスすることによって各 VRAM に切り換わります。

PCG のデータ自体は Z80 のデータ・バスからタイミングをはかって取り込みます。そのタイミングを**図3-17**に示します。

**図3-14 表示されないバイト数**

| モード | ページ | 表示されるアドレス | 表示されないバイト数 |
|---|---|---|---|
| 40字 | 0 | 3000H～33E7H | 24 |
| | 1 | 3400H～37E7H | 24 |
| 80字 | 0 | 3000H～37CFH | 48 |

**図3-15 画面の使用状態**

垂直走査帰線期間の開始がデータ取り込みの開始です。キャラクタは8×8フォント（8バイト）で，Blue, Red, Greenの3色分定義して，はじめて1キャラクタの定義が終了します。よって1キャラクタのPCGを定義するのに24バイトのデータが必要です。

うまくタイミングを取れば，1垂直走査期間で24バイトのデータをすべて定義することも可能です。

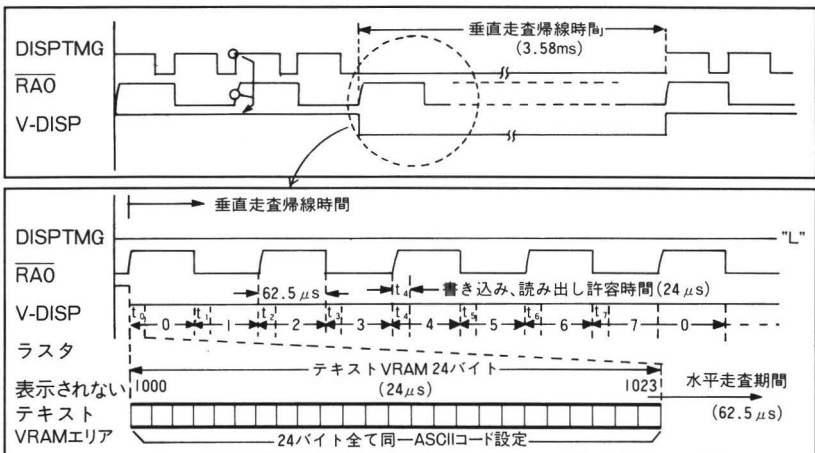
PCG VRAMからキャラクタ・フォント・データを読み出すときも，書き込みと同様にテキストVRAMの未使用のエリアを使って行われます。書き込みとまったく同じタイミングです。

8255②のポートBのビット7が垂直帰線期間信号(V-DISP)の入力端子です。このV-DISP信号がHからLに落ちた時点からが垂直帰線期間の始まりです。

図3 16 **PCG VRAMのI/Oアドレス**

| I/Oアドレス | 内　容 |
|---|---|
| 15＊＊H | PCG VRAM　Blue |
| 16＊＊H | PCG VRAM　Red |
| 17＊＊H | PCG VRAM　Green |

＊＊は無効デジット

図3 17 **PCGへのアクセス・タイミング図(40字モード)**

# 3-2-2 PCG 定義

PCG VRAM をアクセスする前に，テキスト VRAM の未使用エリアに値を書き込んでおかなければならないことは先に説明した通りです。PCG データは 8 ラスタ × 3 色＝24個なので，同じ ASCII コードを24個書き込みます。

```
        LD    BC, 33E8H ; 40字モード，ページ0
        LD    A, 41H ; A＝ASCII コード
        LD    D, 24   ; D＝24バイト分
LOOP:OUT   (C)， A
        INC   BC
        DEC   D
        JR    NZ, LOOP
```

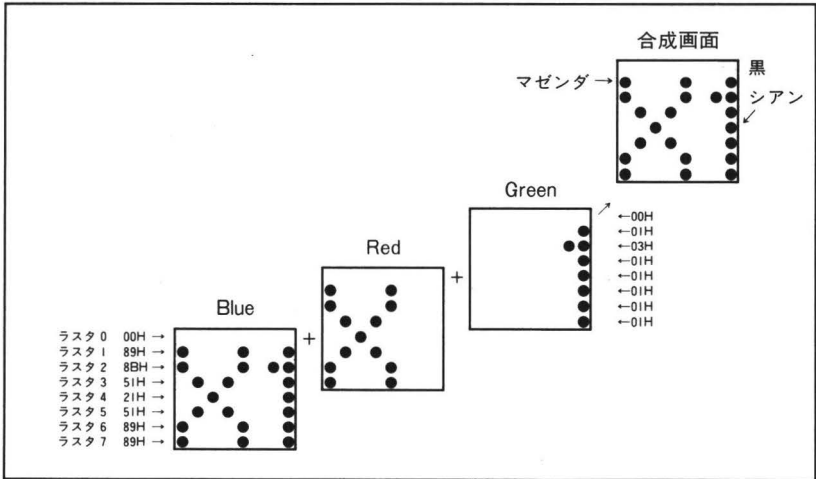
実際に PCG 定義を行う場合は，ディレイ時間管理を厳密に行わなければ正しく定義されません。また，読み出しのときも正確にタイミングを取らなければいけません。V-DISP 信号 1 回で 1 キャラクタ・フォント24バイトすべてを定義するプログラムを**リスト3-6**に示します。また，読み出しのプログラムは**リスト3-7**です。

書き込み，読み出しのプログラムは，IN 命令と OUT 命令が異なるだけでほとんど同じであることがわかると思います。

両リストともに，A レジスタに ASCII コードを，HL レジスタにフォント・データが格納されている（または格納される）先頭アドレスを入れて実行します。

PCG フォント・データはワーク・エリアの先頭から，ラスタ 0 にあたる Blue データ，Red データ，Green データ，次にラスタ 1 にあたる Blue データ，Red データ，Green データ…の順に格納しておきます。

各データは**図3-18**のような横方向のラスタ・データで，3 つの色の合成で 8 色を指定できます。

図3-18 PCGのデータ形式

リスト3-6 No. 1

```
 1:                        ;
 2:                        ;         PCG Write  For 40ｼﾞ ﾓｰﾄﾞ page1
 3:                        ;                                  LIST 3-6
 4:                        ;
 5:  C000                            ORG     0C000H
 6:                        ;
 7:  C000 01E833                     LD      BC,33E8H        ;Text VRAM End
 8:  C003 2153C0                     LD      HL,PCGDAT
 9:  C006 3E41                       LD      A,41H           ;ASCII Code
10:  C008 1618                       LD      D,24
11:  C00A ED79             LOOP1:    OUT     (C),A
12:  C00C 03                         INC     BC
13:  C00D 15                         DEC     D
14:  C00E 20FA                       JR      NZ,LOOP1
15:  C010 3E20                       LD      A,20H
16:  C012 1618                       LD      D,24
17:  C014 01E823                     LD      BC,23E8H
18:  C017 ED79             LOOP2:    OUT     (C),A
19:  C019 03                         INC     BC
20:  C01A 15                         DEC     D
21:  C01B 20FA                       JR      NZ,LOOP2
22:  C01D 010015                     LD      BC,1500H
23:  C020 1E08                       LD      E,08H
24:                        ;
25:  C022 C5                         PUSH    BC
26:  C023 01011A                     LD      BC,1A01H        ;V-DISP
27:  C026 ED78             W1:       IN      A,(C)
28:  C028 F226C0                     JP      P,W1
29:  C02B ED78             W2:       IN      A,(C)
30:  C02D FA2BC0                     JP      M,W2
31:  C030 F3                         DI
32:  C031 C1                         POP     BC
```

```
33:                         ;
34:    C032 7E          LOOP:   LD      A,(HL)
35:    C033 ED79                OUT     (C),A          ;Blue Set
36:    C035 23                  INC     HL
37:    C036 04                  INC     B
38:    C037 7E                  LD      A,(HL)
39:    C038 ED79                OUT     (C),A          ;Red Set
40:    C03A 23                  INC     HL
41:    C03B 04                  INC     B
42:    C03C 7E                  LD      A,(HL)
43:    C03D ED79                OUT     (C),A          ;Green Set
44:    C03F 23                  INC     HL
45:    C040 0615                LD      B,15H
46:                         ;
47:    C042 DD2B                DEC     IX             ;Delay
48:    C044 DD2B                DEC     IX
49:    C046 00                  NOP
50:    C047 3E08                LD      A,08H
51:    C049 3D          W3:     DEC     A
52:    C04A C249C0              JP      NZ,W3
53:                         ;
54:    C04D 1D                  DEC     E
55:    C04E C232C0              JP      NZ,LOOP
56:    C051 FB                  EI
57:    C052 C9                  RET
58:                         ;
59:                         ;               B   R   G
60:    C053 000000      PCGDAT: DEFB    00H,00H,00H
61:    C056 898801              DEFB    89H,88H,01H
62:    C059 8B8803              DEFB    8BH,88H,03H
63:    C05C 515001              DEFB    51H,50H,01H
64:    C05F 212001              DEFB    21H,20H,01H
65:    C062 515001              DEFB    51H,50H,01H
66:    C065 898801              DEFB    89H,88H,01H
67:    C068 898801              DEFB    89H,88H,01H
68:                         ;
69:    C06B                     END
```

リスト3-7

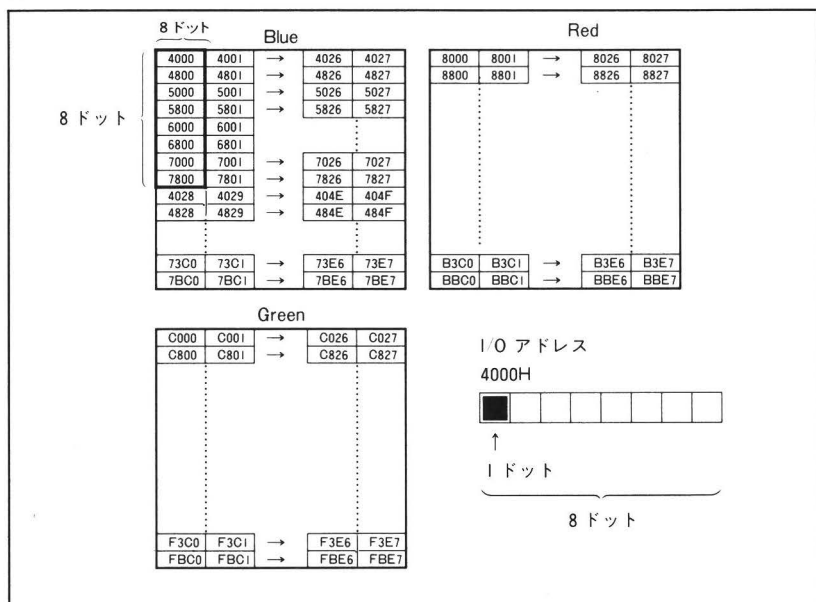
```
 1:                    ;
 2:                    ;        PCG Read
 3:                    ;                                    LIST 3-7
 4:                    ;
 5:  C000                       ORG     0C000H
 6:                    ;
 7:  C000 01E833                LD      BC,33E8H            ;Text VRAM End
 8:  C003 2153C0                LD      HL,PCGDAT
 9:  C006 3E41                  LD      A,41H               ;ASCII Code
10:  C008 1618                  LD      D,24
11:  C00A ED79         LOOP1:   OUT     (C),A
12:  C00C 03                    INC     BC
13:  C00D 15                    DEC     D
14:  C00E 20FA                  JR      NZ,LOOP1
15:  C010 3E20                  LD      A,20H
16:  C012 1618                  LD      D,24
17:  C014 01E823                LD      BC,23E8H
18:  C017 ED79         LOOP2:   OUT     (C),A
19:  C019 03                    INC     BC
20:  C01A 15                    DEC     D
21:  C01B 20FA                  JR      NZ,LOOP2
22:  C01D 010015                LD      BC,1500H
23:  C020 1E08                  LD      E,08H
24:  C022 C5                    PUSH    BC
25:  C023 01011A                LD      BC,1A01H
26:  C026 ED78         W1:      IN      A,(C)
27:  C028 F226C0                JP      P,W1
28:  C02B ED78         W2:      IN      A,(C)
29:  C02D FA2BC0                JP      M,W2
30:  C030 F3                    DI
31:  C031 C1                    POP     BC
32:  C032 ED78         LOOP:    IN      A,(C)               ;Blue Read
33:  C034 77                    LD      (HL),A
34:  C035 23                    INC     HL
35:  C036 04                    INC     B
36:  C037 ED78                  IN      A,(C)               ;Red Read
37:  C039 77                    LD      (HL),A
38:  C03A 23                    INC     HL
39:  C03B 04                    INC     B
40:  C03C ED78                  IN      A,(C)               ;Green Read
41:  C03E 77                    LD      (HL),A
42:  C03F 23                    INC     HL
43:  C040 0615                  LD      B,15H
44:  C042 DD2B                  DEC     IX
45:  C044 DD2B                  DEC     IX
46:  C046 00                    NOP
47:  C047 3E08                  LD      A,08H
48:  C049 3D           W3:      DEC     A
49:  C04A C249C0                JP      NZ,W3
50:  C04D 1D                    DEC     E
51:  C04E C232C0                JP      NZ,LOOP
52:  C051 FB                    EI
53:  C052 C9                    RET
54:                    ;
55:  C053              PCGDAT:  DEFS    24
56:                    ;
57:  C06B                       END
```

# 3-3 グラフィック画面

グラフィック画面は次の2つのモード4種類からいづれか1つを選択することができます。

❶640×200ドット（ドット毎に8色）1画面
❷320×200ドット（　　〃　　）2画面
❸640×200ドット（面単位で8色）　3画面
❹320×200ドット（　　〃　　）　6画面

グラフィックVRAMは，Blue, Red, Greenの3つに分けられており，それぞれ16Kバイトづつあります。グラフィックVRAMの1バイトは画面上で横方向の8ドットに対応しています。そして，グラフィックVRAMは，各モードによって数画面に分割されて使用されます。例として320×200ドット・モードでのI/Oアドレスと画面表示位置との対応を**図3-19**に示します。

**図3-19 I/Oアドレスと表示位置（320×200ドット）**

Blue画面アドレスに4000Hを加えた値がRed画面のアドレスであり，さらに4000Hを加えた値がGreen画面のアドレスとなります。

この3画面を合成することによってドット毎に8色が指定できるわけです。

## 3-3-1 画面モードの切り換え

2つのグラフィック・モード（640×200モードと320×200モード）はテキスト画面のモード（40字モードと80字モード）に対応しており，テキスト画面を切り換えることによって，同時にグラフィック画面のモードも切り換わります。ですから，**リスト3-1**はグラフィック画面を320×200ドットモードに切り換えるプログラムである，ともいえるわけです。

320×200ドット・モードの場合VRAMが2画面分存在するため，ページ0，1の2つに分割して使用しています。ページの切り換えは，CRTCのレジスタ（スタート・アドレス・レジスタのHのみ）を00Hまたは04Hに書き変えることで実行できます(**図3-20**)。

**図3-20 320×200ドット・モードでのページ切換え**

| ページ切り換え | スタート・アドレス(H)レジスタの内容：(R12) |
|---|---|
| 0ページ↔1ページ | 00 H ⟷ 04 H |

ページ0からページ1に切り換えるには次のようなプログラムを使います。

```
LD   BC, 1800H
LD   DE, 0C04H   ;D = 12 (CTRCのレジスタ番号)
                 ;E = 04H (VRAMスタート・
                      アドレス0400H)
OUT  (C), D
INC  B
OUT  (C), E
```

また，２つのページはそれぞれBlue，Red，Greenの３画面の合成で８色を出しています。よって，ページ単位でなく，画面単位で使用した場合，合計６画面となり，パレット・コードを操作して画面単位に８色を指定できます。

たとえば，初期状態から，Blue画面のみ表示する状態に変えるプログラムは次のようになります。

```
LD    BC, 1100H
XOR   A
OUT   (C), A    ; Red 表示せず
INC   B
OUT   (C),A     ; Green 表示せず
```

くわしくは**図3-21**を参考にしてください。

640×200ドットモードの場合，表示ページは１つであり，画面単位では３画面まで使えます。

**図3-21 表示画面とI/Oアドレスおよび設定データの関係**

| 表示画面 ○：表示させる ×：表示させない | | | I/Oアドレスおよび設定データ | | |
|---|---|---|---|---|---|
| GREEN | RED | BLUE | 1200H | 1100H | 1000H |
| × | × | × | 0 0 | 0 0 | 0 0 |
| × | × | ○ | 0 0 | 0 0 | A A |
| × | ○ | × | 0 0 | C C | 0 0 |
| × | ○ | ○ | 0 0 | C C | A A |
| ○ | × | × | F 0 | 0 0 | 0 0 |
| ○ | × | ○ | F 0 | 0 0 | A A |
| ○ | ○ | × | F 0 | C C | 0 0 |
| ○ | ○ | ○ | F 0 | C C | A A |

## 3-3-2 グラフィックによる漢字表示

ワード・プロセッサを使うことが盛んになってきた現在，漢字を画面に表示させるプログラムがなにかと有用になります。

X 1では漢字ROMがオプション（標準装備の機種もあります）で用意されています。この漢字ROMと同一のフォント・データをグラフィック画面に表示させるプログラムを考えてみましょう。漢字ROMからデータを読み出す方法については**3－5**で述べます。

漢字フォント・データは16×16ドット構成で**図3－22**に示すようにラスタ順に並んでいるため比較的簡単に表示することができます。

漢字表示プログラム例が**リスト3－8**です。ここでは，青で，“愛”という漢字を表示しています。

**図3 22** **“愛”のフォント・データ**

リスト38

```
 1:                       ;
 2:                       ;       KANJI
 3:                       ;                                 LIST 3-8
 4:   0007 =              WIDTH0  EQU     0007H
 5:                       ;
 6:   C000                        ORG     0C000H
 7:                       ;
 8:   C000 213DC0                 LD      HL,KDATA
 9:   C003 010040                 LD      BC,4000H
10:                       ;
11:   C006 E5                     PUSH    HL
12:   C007 111000                 LD      DE,0010H
13:   C00A 19                     ADD     HL,DE
14:   C00B EB                     EX      DE,HL
15:   C00C 2600                   LD      H,0
16:   C00E 3A0700                 LD      A,(WIDTH0)
17:   C011 6F                     LD      L,A
18:   C012 09                     ADD     HL,BC
19:   C013 2236C0                 LD      (TD+1),HL
20:   C016 E1                     POP     HL
21:
22:   C017 3E02                   LD      A,2
23:   C019 F5             LOOP1:  PUSH    AF
24:   C01A 3E08                   LD      A,8
25:   C01C F5             LOOP2:  PUSH    AF
26:   C01D 7E                     LD      A,(HL)
27:   C01E ED79                   OUT     (C),A
28:   C020 03                     INC     BC
29:   C021 1A                     LD      A,(DE)
30:   C022 ED79                   OUT     (C),A
31:   C024 0B                     DEC     BC
32:   C025 23                     INC     HL
33:   C026 13                     INC     DE
34:   C027 E5                     PUSH    HL
35:   C028 210008                 LD      HL,0800H
36:   C02B 09                     ADD     HL,BC
37:   C02C 44                     LD      B,H
38:   C02D 4D                     LD      C,L
39:   C02E E1                     POP     HL
40:   C02F F1                     POP     AF
41:   C030 3D                     DEC     A
42:   C031 C21CC0                 JP      NZ,LOOP2
43:   C034 F1                     POP     AF
44:   C035 010000         TD:     LD      BC,0000H
45:   C038 3D                     DEC     A
46:   C039 C219C0                 JP      NZ,LOOP1
47:                       ;
48:   C03C C9                     RET
49:                       ;  --   'AI'  --
50:   C03D 003F1108 KDATA:        DEFB    00H,3FH,11H,08H,7FH,40H,54H,14H
51:   C045 2722070A               DEFB    27H,22H,07H,0AH,31H,00H,07H,78H
52:   C04D 3EC00488               DEFB    3EH,0C0H,04H,88H,0FFH,01H,0C5H,14H
53:   C055 F200FC08               DEFB    0F2H,00H,0FCH,08H,10H,0E0H,18H,07H
54:                       ;
55:   C05D                        END
```

# 3-4 特殊画面制御

Ｘ１ではビデオ信号を制御することによってパレット機能やプライオリティ機能を実現させています。また，Ｘ１がそのさきがけとなったスーパーインポーズ機能もビデオ制御の１つです。

また，I/O アドレスの割り当てを切り換えることによって同時アクセス・モードという特殊なモードをもうけ，高速画面クリアなどを可能にしています。

## 3-4-1 パレット機能

パレット機能とは，アトリビュートやグラフィックVRAM の値を変更することなしに瞬間に表示色を変えることができる機能です。

Ｘ１ではカラーコードを直接 CRT に出力しておらず，途中にパレット選択回路と呼ばれるデータ・セレクタ IC を通しています。通常は，入力されるカラーコードと出力されるパレット・コードが一致しています。

パレット回路は**図 3-23**のようになっていて（$S_1$，$S_2$，$S_3$）がセレクト入力端子，($Y_1$，$Y_2$，$Y_3$)が出力端子で，$Y_1$＝Blue，$Y_2$＝Red，$Y_3$＝Green に割り当てられています。

データ・セレクト端子（$D_0$～$D_7$）には各色のデータ・セレクタ値が初期設定のときにラッチされます。この値を変化させることで入力されたカラーコードは実際の値から変換されて CRT に送り出されます。

各色のデータ･セレクタ値は**図 3-24**のようになっています。

この設定値は，各色のカラーコードが入った場合に$D_0$～$D_7$のうち選択されるビットを“１”にした値であり，内部の結線によって決められています。

３個のデータ･セレクタ IC のデータ・インプット端子（$D_0$

～$D_7$）は図 3 -25の I/O ポートをアクセスすることで Z80と接続されます。

図3-23 **プライオリティ回路とパレット回路のブロック図**

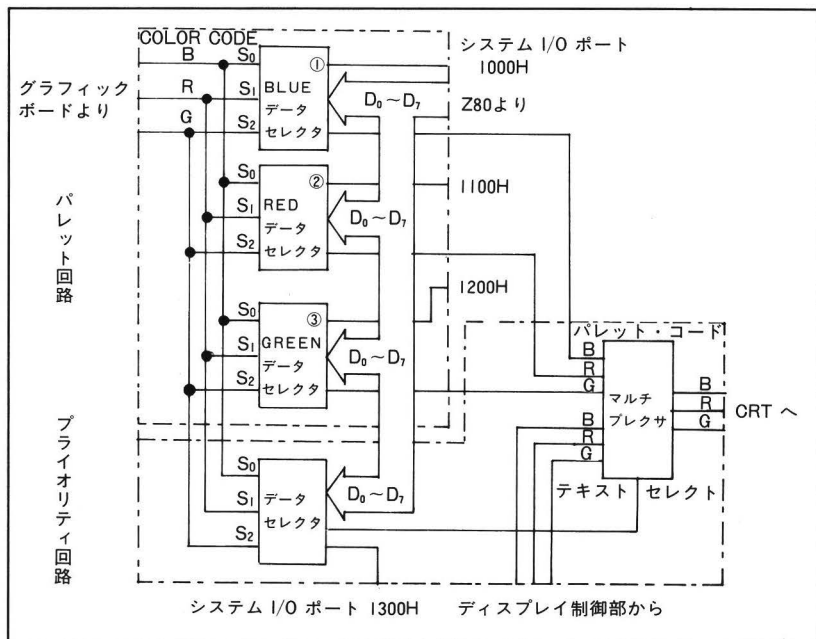



図3-24 **データ・セレクタ値**

| | |
|---|---|
| Blue データ・セレクタ | AAH |
| Red データ・セレクタ | CCH |
| Green データ・セレクタ | F0H |

図3-25 **データ・セレクタと I/O アドレス**

| データ・セレクタ IC | I/O アドレス |
|---|---|
| Blue | 10**H |
| Red | 11**H |
| Green | 12**H |

**は無効デジット

パレットにカラーコードを設定するプログラム例を**リスト3-9**に示します。

データ・インプット端子を選択する場合にはかならずOUT命令で行ってください。IN命令で読み出そうとすると画面全体が白くなってしまいます。つまり，パレット回路の設定値は読み出すことができないということです。よって，あらかじめパレット用のワーク・エリアをメモリ上に3バイト用意しておき，書き込むときは双方に書き込み，読み出すときはワーク・エリアから読み出すようにします。

リスト3-9 No. 1

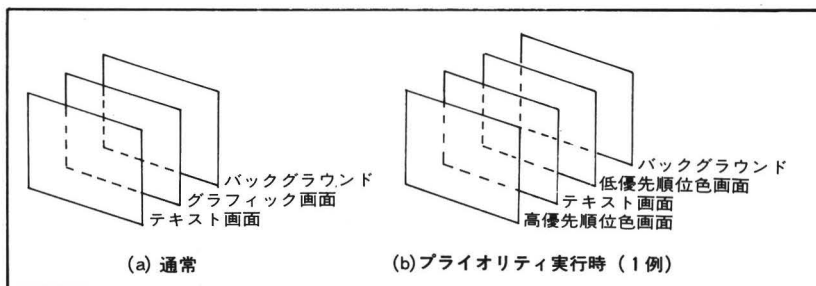
```
 1:                  ;
 2:                  ;       Pallet Set
 3:                  ;                                 LIST 3-9
 4:                  ;
 5:  C000                    ORG     0C000H
 6:                  ;
 7:  C000 1601               LD      D,1               ;Color Code
 8:  C002 1E00               LD      E,0               ;Pallet NO.
 9:                  ;
10:  C004 212EC0             LD      HL,PAL0
11:  C007 7A                 LD      A,D
12:  C008 1600               LD      D,0
13:  C00A 19                 ADD     HL,DE
14:  C00B 77                 LD      (HL),A
15:  C00C 2135C0             LD      HL,PAL7
16:  C00F 0608               LD      B,8
17:  C011 110000             LD      DE,0000
18:  C014 0E00               LD      C,00
19:  C016 7E         LOOP:   LD      A,(HL)
20:  C017 1F                 RRA
21:  C018 CB13               RL      E                 ;Blue
22:  C01A 1F                 RRA
23:  C01B CB12               RL      D                 ;Red
24:  C01D 1F                 RRA
25:  C01E CB11               RL      C                 ;Green
26:  C020 2B                 DEC     HL
27:  C021 10F3               DJNZ    LOOP
28:  C023 0610               LD      B,10H
29:  C025 ED59               OUT     (C),E
30:  C027 04                 INC     B
31:  C028 ED51               OUT     (C),D
32:  C02A 04                 INC     B
33:  C02B ED49               OUT     (C),C
34:  C02D C9                 RET
35:                  ;
36:  C02E 00         PAL0:   DEFB    0
37:  C02F 00         PAL1:   DEFB    0
38:  C030 00         PAL2:   DEFB    0
39:  C031 00         PAL3:   DEFB    0
40:  C032 00         PAL4:   DEFB    0
```

No. 2

```
41:   C033 00       PAL5:   DEFB    0
42:   C034 00       PAL6:   DEFB    0
43:   C035 00       PAL7:   DEFB    0
44:                 ;
45:   C036                  END
```

# 3-4-2 プライオリティ機能

プライオリティ機能とは，テキスト画面とグラフィック画面の重なりにおいて，各色の間に優先順位を持たせることによって奥行きを表現することができる機能です。通常，グラフィック画面の方がテキスト画面の後方に位置しますが，プライオリティ機能を使うことによってグラフィック画面を前方に引き出すことができるわけです（**図 3 -26**）。

**図3 26 プライオリティ機能**

また，プライオリティは色単位に設定できるので，テキスト画面を基準に前後に奥行きをもたせることができます。さらにパレット機能との組み合わせによってより立体的な画面表示が可能です。

プライオリティ回路は**図 3 -23**にあるようにパレット回路とCRT入力端子の中間にあり，マルチプレクサICで構成されています。

マルチプレクサICにはテキストVRAM出力のB・R・G信号と，グラフィックVRAM出力からパレット回路を通し

たB·R·G信号が入力されており，セレクト信号によってどの色のビット内容を出力するか決定しています。

セレクト信号はグラフィックVRAMのカラー制御ビットの組み合わせで設定され，セレクトがLのときテキストVRAM出力B·R·G信号を出力し，セレクトがHのときグラフィックVRAM出力B・R・G信号を出力します。

テキストVRAMからの出力データがない場合は当然グラフィックVRAM出力B・R・G信号が優先されます。

プライオリティを設定するには，プライオリティ回路のデータ・セレクタICに値を書き込むことで実現できます。

初期設定値として00Hがラッチされているので**図3-26(a)**のようにテキスト画面が，グラフィック画面，バックグラウンド画面のいずれの色よりも優先され，もっとも手前に見えます。

優先順位の組み合わせは**図3-27**のようになっています。

プライオリティ回路のデータ・セレクタICのチップ・セレクトはI/Oポートの13＊＊Hに割りあてられており，優先順位を設定する場合はI/Oポートの13＊＊Hに値を書き込むことで実行できます。＊＊は無効デジットであり，どんな値でもかまいません。

図3-27 **優先順位**

**プライオリティのコード内容**

| ビット | 内　容 |
|---|---|
| $D_0$ | 黒（バック色）の優先 |
| $D_1$ | 青の優先 |
| $D_2$ | 赤の優先 |
| $D_3$ | マゼンタの優先 |
| $D_4$ | 緑の優先 |
| $D_5$ | シアンの優先 |
| $D_6$ | 黄色の優先 |
| $D_7$ | 白の優先 |

**例**

| $D_7$ | $D_6$ | $D_5$ | $D_4$ | $D_3$ | $D_2$ | $D_1$ | $D_0$ | |
|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | ＝14H |

赤と緑のグラフィック色がテキストより優先される。

各ビットが1のときその色がテキストより優先される。

例として，赤と緑をTEXTより優先させるプログラムを次に示します。

```
LD   B, 13H
LD   A, 14H
OUT  (C), A
```

## 3-4-3 スーパーインポーズ機能

スーパーインポーズ機能とは，一般のテレビ放送画面やVTR等のビデオ画面とパソコンの画面を重ね合わせる機能です。

X 1の画面の水平・垂直同期信号はCRTCにより制御されており，この信号に同期を取りながらうまくパソコン画像信号を出力すればスーパーインポーズが実現できるわけです。

日本のテレビ放送の方式はNTSC方式と呼ばれていますが，X 1ではそのNTSC方式よりも短かめに周期を設定しています。その周期の不足時間はCRTCの入力カウントを停止させて同期をとり，同期信号の補正は1周期ごとに行なわれます。

スーパーインポーズ開始の命令はZ80からサブCPU80C49にコードを送ることによって行われます。

モニタ画面には次の4通りのモードがあります。

**1** TV画面

**2** コンピュータ画面

**3** スーパーインポーズ画面1（コンピュータのコントラストをダウンさせる）

**4** スーパーインポーズ画面2（テレビのコントラストをダウンさせる）

これらの4つのモードを切り換えるには，まず画面切り換えの指示コードである。"E7H"を送ります。すると，80C49は次に画面切り換えコードがくるものと判断します。"E7H"につづけて特定のコードを送ることによって各モードに切り

換わります。

各モードに対するコード列を**図3-28**に示します。**リスト3-10**は画面モード切り換えのプログラム例です。DATAの部分にそれぞれの送信コード列をセットしてこのプログラムを実行するとモードを換えることができます。このプログラムではコードの終わりのしるしとして最後に00Hを入れています。

この例ではスーパーインポーズ画面2に設定されます。

**図3-28 モードの設定コード**

| 画面モード ＼ 送信コード | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|
| TV画面 | E7 | 05 | | | | |
| コンピュータ画面 | E7 | 05 | E7 | 08 | | |
| スーパーインポーズ1（コントラスト・ダウン） | E7 | 05 | E7 | 0F | E7 | 0A |
| スーパーインポーズ2（コントラスト・ノーマル） | E7 | 05 | E7 | 0F | | |

**リスト3-10**

```
 1:                    ;
 2:                    ;         Superimpose Mode 2
 3:                    ;                                  LIST 3-10
 4:   0B54 =           TRNS49    EQU      0B54H
 5:                    ;
 6:   C000                       ORG      0C000H
 7:                    ;
 8:   C000 210DC0                LD       HL,DATA
 9:   C003 7E          LOOP:     LD       A,(HL)
10:   C004 A7                    AND      A
11:   C005 C8                    RET      Z
12:   C006 CD540B                CALL     TRNS49
13:   C009 23                    INC      HL
14:   C00A C303C0                JP       LOOP
15:                    ;
16:   C00D E705E70F    DATA:     DEFB     0E7H,05H,0E7H,0FH
17:   C011 00                    DEFB     00H                        ;Data End
18:                    ;
19:   C012                       END
```

## 3-4-4 同時アクセス・モード

X 1の表示画面には，テキスト画面，グラフィック画面，バックグラウンドの3種類あります。

テキストVRAM，グラフィックVRAMともにI/Oアドレスに割りあてられており，電源投入時は**図3-29(a)**のようになっています。

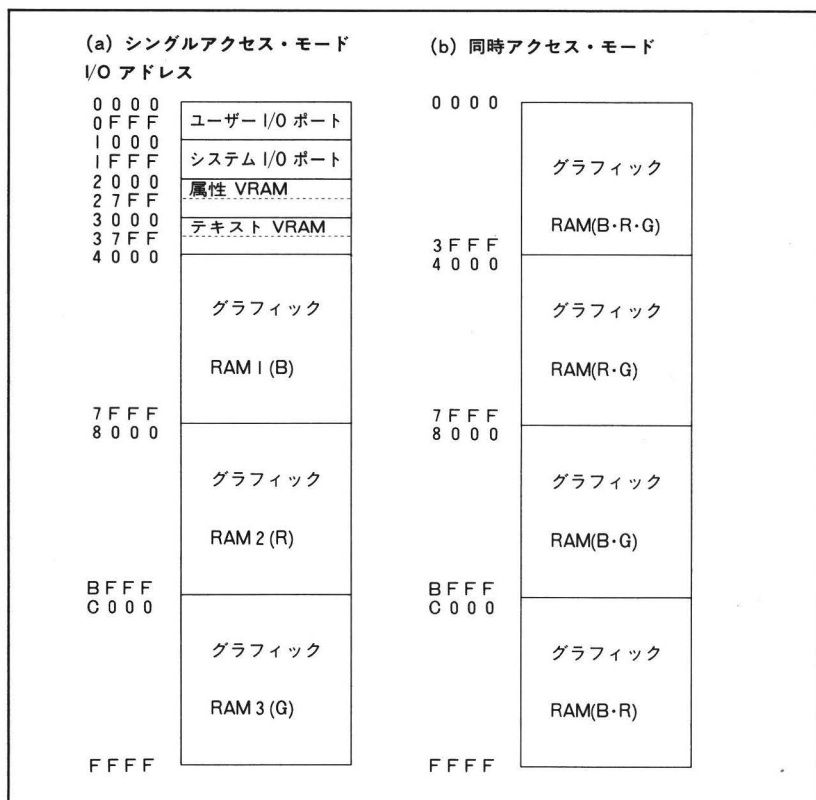
グラフィックVRAMは，同時アクセス・モードに切り換えることができ，そのときのI/Oマップは**図3-29(b)**です。

**図3-29 I/O マップ**

(a) シングルアクセス・モード

| I/O アドレス | |
|---|---|
| 0000<br>0FFF | ユーザー I/O ポート |
| 1000<br>1FFF | システム I/O ポート |
| 2000<br>27FF | 属性 VRAM |
| 3000<br>37FF | テキスト VRAM |
| 4000<br>7FFF | グラフィック RAM 1 (B) |
| 8000<br>BFFF | グラフィック RAM 2 (R) |
| C000<br>FFFF | グラフィック RAM 3 (G) |

(b) 同時アクセス・モード

| I/O アドレス | |
|---|---|
| 0000<br>3FFF | グラフィック RAM(B・R・G) |
| 4000<br>7FFF | グラフィック RAM(R・G) |
| 8000<br>BFFF | グラフィック RAM(B・G) |
| C000<br>FFFF | グラフィック RAM(B・R) |

同時アクセス・モードではI/Oアドレスがすべてグラフィック VRAM にあてられていて，1バイトのアクセスで実際は複数のグラフィック VRAM をアクセスしたことになります。よって，画面のクリアや塗りつぶしなどで高速処理が実行できます。

同時アクセス・モードに切り換えるには，まずダミーでIN命令を行います。

ここで注意しなければならないのは，ダミーのI/Oポートの読み込みのときにBCレジスタの値がパレットのI/Oアドレスになっていてはいけない点です。パレット機能のところでも説明しましたが，こうすると画面が真白になってしまいます。

よって，あらかじめBレジスタに10H，11H，12H以外の値を入れておくことと良いでしょう。

なぜ，ダミーのIN命令を入れるのかというと，現在すでに同時アクセス・モードになっている状態でさらに同時アクセス・モードに変換しようとした場合，ダミーがないと正しく変換されないためです。

8255②のポートCのビット5がアクセス・モードの切り換え端子（切り換え用フリップフロップのCK端子）につながっており，ここに1クロック・パルスを送ることで切り換わります。

また，切り換え用のフリップフロップのクリア端子に$\overline{\text{IORD}}$信号が入っています。つまりI/O空間を1度でも読み出すことによってシングルアクセス・モードに戻るわけです。

同時アクセス・モードにかえるプログラムを**リスト3-11**に示します。

リスト3-11

```
 1:                     ;
 2:                     ;          Simultaneous Access & Graphic Clear
 3:                     ;                                 LIST 3-11
 4:                     ;
 5:  C000                          ORG     0C000H
 6:                     ;
 7:  C000 0617                     LD      B,17H          ;---ﾄﾞｳｼﾞ Access Mode
 8:  C002 ED78                     IN      A,(C)          ;Dummy
 9:  C004 01031A                   LD      BC,1A03H
10:  C007 3E0B                     LD      A,0BH
11:  C009 ED79                     OUT     (C),A
12:  C00B 3D                       DEC     A
13:  C00C ED79                     OUT     (C),A
14:                     ;
15:  C00E F3                       DI
16:  C00F 010040                   LD      BC,4000H       ;---Clear
17:  C012 1600                     LD      D,0
18:  C014 0B            LOOP:      DEC     BC
19:  C015 ED51                     OUT     (C),D
20:  C017 0B                       DEC     BC
21:  C018 ED51                     OUT     (C),D
22:  C01A 78                       LD      A,B
23:  C01B B1                       OR      C
24:  C01C C214C0                   JP      NZ,LOOP
25:  C01F FB                       EI
26:                     ;
27:  C020 0617                     LD      B,17H          ;---Single Access Mode
28:  C022 ED78                     IN      A,(C)
29:  C024 C9                       RET
30:                     ;
31:  C025                          END
```

# 3-5 漢字フォントの読み出し

X 1は，漢字ROM（CZ-8KR）を取り付けることにより，容易に漢字表示を行えます。

この漢字フォントの読み出し，表示をBASICではKANJI$( ), POSITION, PATTERNで行います。

たとえば，左上すみに“漢”を表示する場合，

```
A$ = KANJI$(&H3441)
POSITION 0,0
PATTERN −16, A$
```

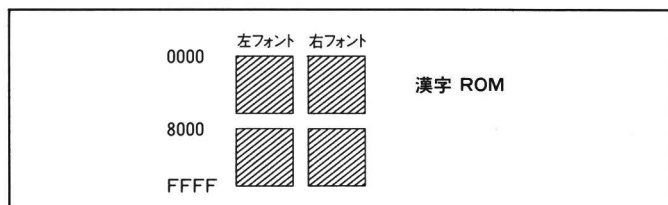
とします。ここでは，このKANJI$という漢字ROMのフォント・データを読むマシン語サブルーチンを紹介します。

## 3-5-1 漢字ROMの構成

漢字ROMは，4個のマスクROMと4個のソケットで構成されています。

マスクROMは，256Kビットで構成されています。4個のソケットは，カナ漢字変換ROMを入れたり，C-MOS RAMを入れて自分で作ったフォントを登録するときのために設けてあります。

漢字ROMをZ80からみたアドレス構成は，**図3-30**のようになります。

**図3-30 漢字ROMの構成**

漢字ROMは，ワード構成になっていて，左フォント，右フォントが別々に登録されています。

片側256Kビット2個で構成されるので，そのアドレスは，0000H～FFFFHとなります。これが，ワード構成なので，128Kバイトになります。

漢字ROMのI/Oアドレスは，**図3-31**に示すように0E80H～0E82Hです。

**図3-31 漢字ROMのI/Oアドレス**

| I/Oポート | 操作内容 |
|---|---|
| 0E80 | 1．収納アドレス下位データ設定<br>2．左側データ読み込みポート |
| 0E81 | 1．収納アドレス上位データ設定<br>2．右側データ読み込みポート |
| 0E82 | 1．チップ・セレクト ON [0E82H←01]<br>2．チップ・セレクト OFF [0E82H←00] |

# 3-5-2 アドレス算出

JISコードは，そのまま順番になっているのではなく，とびとびになっています。このため，漢字フォントのアドレスを求めるためには，簡単な計算が必要です。

その計算は，次のようになります。

**❶区点コード→JISコード変換**

区点コードからJISコードに変換します。これには，

| JISコード上位バイト＝（区）H＋20H |
|---|
| 〃　下位　〃　＝（点）H＋20H |

とします。

**❷JISコード→収納アドレス**

❶で求まったJISコードから収納アドレスを計算します。これには，次の方法で計算します。

● JIS コードの上位バイト＞28H の場合

| ((JIS コード上位バイト－30H)×６)×100H＋4000H |
|---|

● JIS コードの上位バイト≦28H の場合

| ((JIS コード上位バイト－21H)×６)×100H＋100H |
|---|

最後に，

| 上式で求まった値＋（JIS コード下位バイト－20H)×10H |
|---|

で収納アドレスが求まります。

例として“漢”のアドレスを求めてみましょう。“漢”の区点は2033なので，

| 区＝20（14H）<br>点＝33（21H） |
|---|

です。次に，

| JIS コード上位＝（区）H＋20H＝14H＋20H＝34H<br>〃　　下位＝（点）H＋20H＝21H＋20H＝41H |
|---|

となり，JIS コードの上位バイトは28H より大きいので，

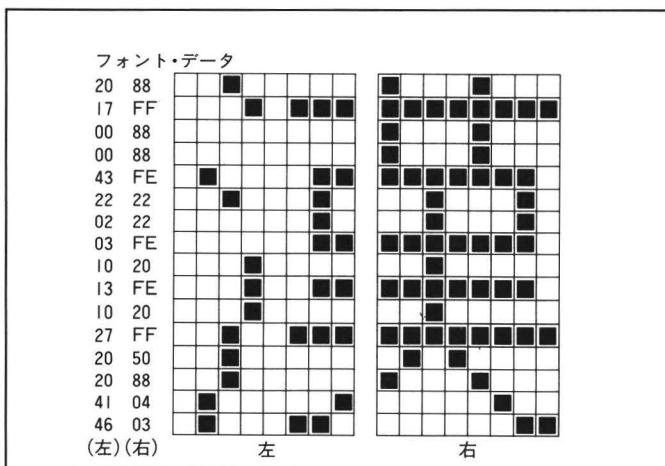
| ((34H－30H)×６)×100H＋4000H＝5800H |
|---|

となります。したがって収納アドレスは，

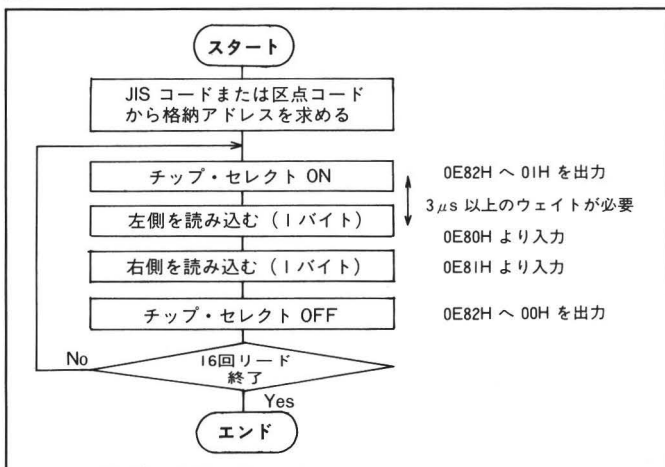
| 5800H＋（41H－20H)×10H＝5A10H |
|---|

と求まります。 5A10H から32バイトが“漢”のフォント・データです。

漢字フォントの構成は，図３-32に示すようになっています。

図3 32 “漢”のフォント

# 3-5-3 フォントを読み出すプログラム

JIS コードあるいは区点コードより，格納アドレスが求まったら，次はI/O ポートをアクセスしてフォント・データを読み出します。このアルゴリズムは図 3 -33のようになり，プ

図3 33 漢字フォントのリード・アルゴリズム

ログラムは **リスト3-12** のようになります。このプログラムは，'KCODE'に JIS コード，'KDATA'にフォント・データを格納する先頭アドレスを入れてコールします。サンプルを**リスト3-13** に示します。

リスト3-12

No. 1

```
 1:                    ;
 2:                    ;        KANJI  IN
 3:                    ;                                LIST 3-12
 4:                    ;
 5:  F000                       ORG      0F000H
 6:                    ;
 7:  F000 C307F0      KANJI:    JP       KANJE
 8:                    ;
 9:  F003 00          KCODE:    DEFB     00            ;jis code low
10:  F004 00                    DEFB     00            ;         high
11:  F005 0000        KDATA:    DEFW     0000          ;read adr.
12:                    ;
13:  F007 F5          KANJE:    PUSH     AF            ;save reg.
14:  F008 C5                    PUSH     BC
15:  F009 D5                    PUSH     DE
16:  F00A E5                    PUSH     HL
17:                    ;
18:  F00B ED4B03F0              LD       BC,(KCODE)    ;read code
19:  F00F 211930                LD       HL,3019H      ;high code>28h
20:  F012 B7                    OR       A             ;CF=0
21:  F013 ED42                  SBC      HL,BC         ;
22:  F015 3805                  JR       C,KCHR        ;if high code<=28h
23:  F017 110121                LD       DE,02101H     ;D=21h offset1
24:                                                    ;E=01h offset2
25:  F01A 1803                  JR       KJAD
26:  F01C 114030      KCHR:     LD       DE,3040H      ;D=30h offset1
27:                                                    ;E=40h offset2
28:  F01F 78          KJAD:     LD       A,B           ;high code
29:  F020 92                    SUB      D             ;sub offset1
30:  F021 87                    ADD      A,A           ;*2
31:  F022 47                    LD       B,A
32:  F023 87                    ADD      A,A           ;*4
33:  F024 80                    ADD      A,B           ;*6
34:  F025 83                    ADD      A,E           ;add offset2
35:  F026 57                    LD       D,A           ;
36:  F027 1E00                  LD       E,00
37:                    ;
38:  F029 79                    LD       A,C           ;low code
39:  F02A D620                  SUB      20H           ;jis --> TEN
40:  F02C 2600                  LD       H,00          ;
41:  F02E 6F                    LD       L,A           ;ld l,e
42:  F02F 29                    ADD      HL,HL         ;*2
43:  F030 29                    ADD      HL,HL         ;*4
44:  F031 29                    ADD      HL,HL         ;*6
45:  F032 29                    ADD      HL,HL         ;*16
46:  F033 19                    ADD      HL,DE         ;start adr. of font
47:                             ;
48:  F034 ED5B05F0 KREAD:       LD       DE,(KDATA)    ;read adr.
49:  F038 EB                    EX       DE,HL         ;change DE and HL
50:                             ;
```

```
51:  F039 01800E           LD    BC,0E80H     ;rom adr.
52:  F03C ED59             OUT   (C),E        ;font low adr. set
53:  F03E 59               LD    E,C          ;copy rom low adr. E=80h
54:  F03F 0C               INC   C            ;rom adr.  0E81h
55:  F040 ED51             OUT   (C),D        ;font high adr. set
56:  F042 0C               INC   C            ;rom adr.  0E82h
57:  F043 1610             LD    D,16         ;loop count
58:                 ;
59:  F045 3E01      KNJRD: LD    A,01H        ;chip sel. on
60:  F047 ED79             OUT   (C),A        ;          0E82h
61:  F049 00               NOP                ;wait
62:                 ;
63:  F04A 4B               LD    C,E          ;C=80h     0E80h
64:  F04B ED78             IN    A,(C)        ;read left font
65:  F04D 77               LD    (HL),A       ;store memory
66:  F04E 23               INC   HL
67:  F04F 0C               INC   C            ;C=81h     0E81h
68:  F050 ED78             IN    A,(C)        ;read right font
69:  F052 77               LD    (HL),A       ;store memory
70:  F053 23               INC   HL
71:                 ;
72:  F054 0C               INC   C            ;C=82h     0E82h
73:  F055 AF               XOR   A            ;chip sel. off
74:  F056 ED79             OUT   (C),A        ;          0E82h
75:  F058 15               DEC   D            ;count down
76:  F059 20EA             JR    NZ,KNJRD     ;if D=00
77:                 ;
78:  F05B E1               POP   HL           ;load reg.
79:  F05C D1               POP   DE
80:  F05D C1               POP   BC
81:  F05E F1               POP   AF
82:  F05F C9               RET                ;return to main
83:                 ;
84:  F060                  END
```

リスト3-13

```
1000 FONT=&H3441       :' KANJI KODE (KCODE)
1010 ADR =&HF080       :' FONT  DATA SET ADR (KDATA)
1020 POKE &HF003,FONT MOD 256 : POKE &HF004,FONT ¥ 256
1030 POKE &HF005,ADR  MOD 256 : POKE &HF006,ADR  ¥ 256
1040 CALL &HF000       :' CALL KANJI
1050 END
```

# 第4章

## 周辺チップ



X1は，テレビ，キー入力，タイマーコントロール，カセット・コントロールなど複雑な処理を80C49に行わせています。これらの処理は80C49に定められたコマンドを送って行うのですが，Z80から直接送るのではなく8255を介して送っています。この8255こそX1を操作する上で非常に重要なチップと言えます。本章では8255の使い方と80C49のコマンドについて述べていきます。

# 4-1 8255

8255はプログラマブル周辺インターフェイス(Programmable Peripheral Interface. PPIと略称される）と呼ばれる1チップのLSIです。この名前が示すとおり，CPUと周辺機器を仲介するチップとして80系のCPUに非常によく使われています。

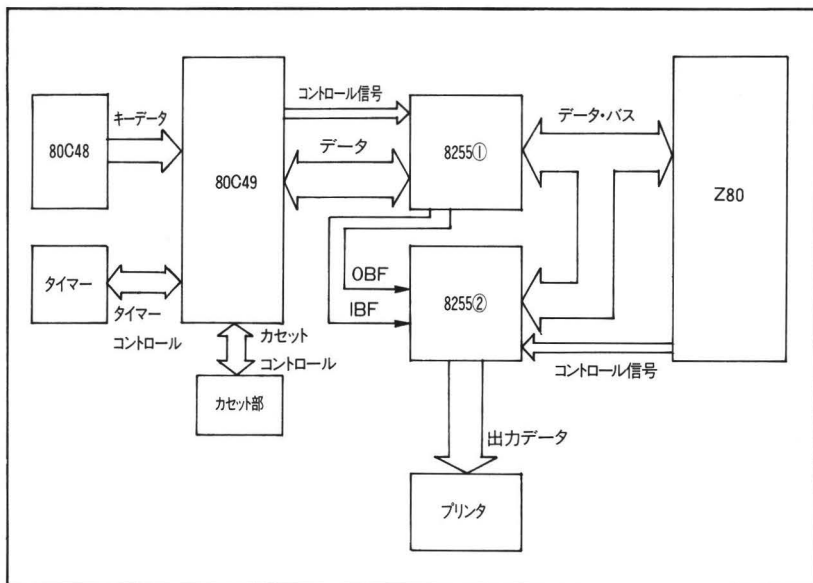
8255は3組の8ビット並列ポートを持ちそれぞれのポートを，ポートA，ポートB，ポートCと呼んでいます。さらに，これらのポートの動作モード（入力か出力あるいは入出力）やポートCのビット・セット，リセットを行う8ビットのコントロール・レジスタがあります。動作モードはモード0,1,2の3種類です。

## 4-1-1 ブロック図

X1では2個の8255を使っています。**図4-1**は X1における Z80, 8255, 80C49のブロック図です。この図のコントロール信号（コントロール・レジスタ）に注目してください。一方はZ80から，他方は80C49から出ています。このことはZ80から8255①のモードを決めることはできないということを表しています。以後，混乱しないように，この2個の8255を

| | | |
|---|---|---|
| 80C49 | 側8255 | ———8255① |
| Z80 | 側8255 | ———8255② |

と区別することにします。

図4 1 **ブロック図**

# 4-1-2 I/O ポートと コントロール・レジスタ

8255は，コントロール・レジスタに書き込む値によって次のような3つのモードを選択できます。

| MODE0 | 基本的な入出力 |
|---|---|
| MODE1 | 制御信号を用いて行う入出力 |
| MODE2 | 双方向データを扱う入出力 |

8255②は，HuBASIC ではモード0でポートAを出力，ポートBを入力，ポートCを出力に設定しています。このモードの設定，入出力の方向は，ハードウェアと密接な関係があり,ユーザーが勝手に変えることは避けた方がよいでしょう。ここでは8255についての詳しい解説はしないので，興味のある方は他の文献を参考にしてください。また,8255①は 80C49

によって設定されているので，ユーザーが変更することはできません。

8255に関するI/Oポートを**表4-1**に示します。ポートの1A03Hには8255②のコントロール・レジスタがありますが，これはモード選択の他にポートCの任意のビットをセットあるいはリセットすることができます。**図4-2**にコントロール・レジスタのビット構成を示します。たとえば，ポートCのビット6(40字/80字モード切り換え)を1(40字モード)にする場合，

```
LD    BC, 1A02H    ;8255②ポートC
IN    A, (C)
SET   6, A
OUT   (C), A
```

と書けますが，コントロール・レジスタを使うと，

```
LD    BC, 1A03H    ;8255②コントロール・レジスタ
LD    A, 0DH       ;00001101B
OUT   (C), A
```

とも書けます。なお，実際に画面モードを切り換えるには，CRTCの設定データも変えなければなりません。また，ポートC(I/Oアドレス1A02H)は出力のみとなっていますが，IN命令で読み込んでもまったく問題はありません。

**表4-1 8255に関するI/Oポート**

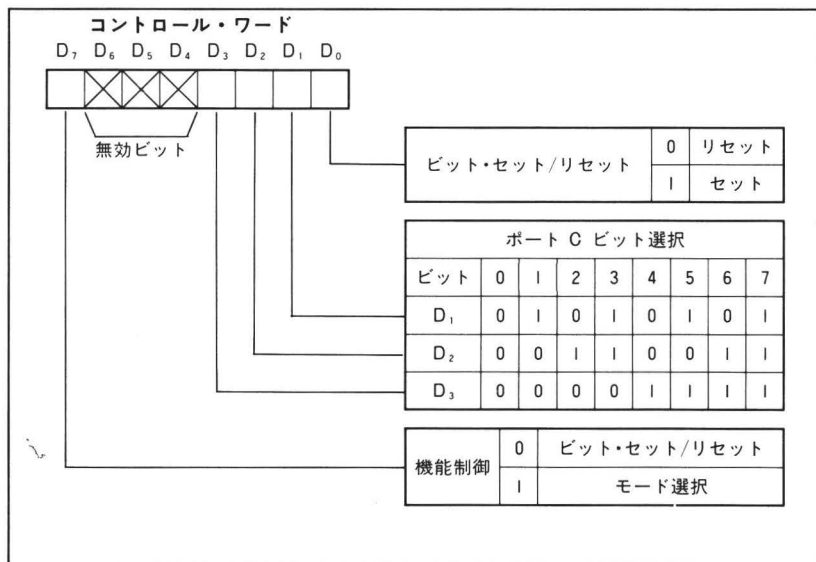
| I/O アドレス | 内　　容 |
|---|---|
| 19＊＊ | 8255①ポート A（80C49 との入出力） |
| 1A00 | 8255②ポート A（プリンタ出力） |
| 1A01 | 〃　〃　B（各種制御信号） |
| 1A02 | 〃　〃　C（　〃　） |
| 1A03 | 〃　コントロール・レジスタ |

●1A01（ポート B）詳細

| ビット | 内　　容 |
|---|---|
| 7 | 垂直帰線信号 |
| 6 | 80C49 へのデータ転送禁止信号　(IBF) |
| 5 | 80C49 からのデータ受け取り可能信号　(OBF) |
| 4 | ――― (XID ではディスクからの生データ) |
| 3 | プリンタからの入力可能信号　(BUSY) |
| 2 | 垂直同期信号 |
| 1 | カセット読み出しデータ |
| 0 | SHIFT ＋ BREAK（キーボード）信号 |

●1A02（ポートC）詳細

| ビット | 内　　容 |
|---|---|
| 7 | 立ち上がりでプリンタは入力データをサンプル |
| 6 | 40字／80字モード（40字：1，80字：0）クロック切り換え |
| 5 | I/O アクセス・モード切り換え（シングル／同時） |
| 4 | スムーズ・スクロール |
| 3～1 | ―――――― |
| 0 | カセットへの書き込みデータ |

図4 2 **コントロール・ワード**

# 4-2 80C49

80C49，80C48はシングルチップ・マイクロコンピュータで，Z80の負担を軽くするために用いられています。80C49はキーボード（80C48）からのデータ入力，クロックICのコントロール，カセットテープデッキのコントロール，テレビのコントロールを直接または8255①を介して制御しています。

## 4-2-1 Z80と80C49のコミュニケーション

80C49の機能を使うために，Z80は80C49に対して一定のコマンドを送らなければなりません。しかし，**図4-1**で見たように，Z80と80C49は直接つながっているわけではなく，途中に8255①があり，コマンドを送るためには8255①のポートA（I/Oアドレス1900H）を仲介します。ただし，コマンドは単純に送れるわけではなく，**図4-1**に示したIBF（Input Buffer Full）とOBF（Output Bufer Full）を監視して行わなければなりません。これらは，それぞれI/Oアドレス1A01Hの6ビット目と5ビット目に現れます。

● **Z80→80C49**

Z80から80C49へデータを送る場合，Z80は8255②のポートBのIBF（ビット6）が0になるのを待って，I/Oアドレス19＊＊Hにデータを出力します。ですから，1バイトのデータを送る場合，次のようになります。

```
      LD   BC, 1A01H ;8255②ポートB
LOOP:IN   A, (C)     ;┐
      BIT  6, A       ;│  IBFが0になるまでループ
      JR   NZ, LOOP;┘
```

```
;
        LD     BC, 19＊＊H；＊＊はどんな値でもよい
        LD     A, データ
        OUT    (C), A
```

● 80C49→Z80

Z80 が 80C49 よりデータを取り込む場合も同様に，今度は 8255②のポート B の OBF（ビット 5）が 0 になるのを待って，I/O アドレス19＊＊H からデータを取り込みます。プログラムは次のようになります。

```
        LD   BC, 1A01H ;8255②ポートB
LOOP:   IN   A, (C)     ; ┐
        BIT  5, A       ; │ OBF が 0 になるまでループ
        JR   NZ, LOOP;    ┘
;
        LD   BC, 19＊＊H
        IN   A, (C)
```

# 4-3 送信要求コード

80C49に対するコマンドは，全部で27個あります(**表4-2**)。コマンドは**4-2**で述べたような方法で送りますが，IOCSにはこのためのサブルーチンが用意されており，この使い方は次のようになっています。

- ・TRANS49　(0B54H)
  機能…Aレジスタの内容を80C49に送る
- ・RECV49　(0B49H)
  機能…80C49からAレジスタにデータを受け取る

以下の説明では，このサブルーチンを使っていきます。

**表4-2 送信要求コード**

| 送信要求コード | 内容 | 後続バイト数 | 方向 80C49 Z80 | 説明 |
|---|---|---|---|---|
| E4 | キーベクタ値をセット | 1 | ← | Z80のキーのベクターアドレスの下位バイトを返す。ただし，0の場合にはキー割り込み禁止モードとなる。 |
| E6 | キーバッファ読み出し | 2 | ← | 80C49のキーバッファの内容をZ80へ送る。キーバッファには最新のデータが格納されている。 |
| E7 | テレビ・コントロール | 1～5 | ← | 各種テレビのコントロール |
| E8 | テレビ送信コード読み出し | 1 | → | テレビに最後に送られたコードをZ80へ返す。 |
| E9 | カセット指示 | 1 | ← | カセットメカの動作をする。 |
| EA | カセット状態読み出し | 1 | → | カセットメカの動作状態を読み出しZ80へ返す。 |
| EB | カセットセンサー読み出し | 1 | → | カセットセンサーを読み，その状態をZ80へ返す。 |
| EC | 日付けセット | 3 | ← | タイマー用ICに“月”，“日”，“曜日”のデータを書き込む。 |
| ED | 日付け読み出し | 3 | → | タイマー用ICから“月”，“日”，“曜日”のデータを読み出しZ80へ返す。 |
| EE | 時刻セット | 3 | ← | タイマー用ICに“時”，“分”，“秒”のデータを書き込む。 |
| EF | 時刻読み出し | 3 | → | タイマー用ICから“時”，“分”，“秒”のデータを読み出す。 |
| D0～D7 | テレビ・タイマー0，1，2，3，4，5，6，7をセット（計8個） | 6 | ← | タイマー0，1，2，3，4，5，6，7の領域にデータを設定する。 |
| D8～DF | テレビ・タイマー0，1，2，3，4，5，6，7を読み出す。 | 6 | → | タイマー0，1，2，3，4，5，6，7の領域からデータを読み出す。 |

## 4-3-1 キー入力と割り込み

X1のキー入力は80C49, 80C48によってコントロールされます。この手順は次のようになっています。

❶キー入力をすると，キーボードの中にある80C48によってスキャンされ，ファンクション・コードとASCIIコードの2バイトに変換します。

❷このデータを80C48がシリアルに変換し80C49に送ります。

❸ 80C49が，バイトに変換してZ80に送ります。

このキーデータの取り込みには割り込みによるものとよらないものの2種類があります。

Z80には，大きく分けて2つの割り込みがあります。ひとつはソフトウェアで禁止（DI命令）や解除（EI命令）ができるもので，キー入力割り込みもこれに含まれます。

もうひとつは，ソフトウェアでは禁止できない割り込みで，NMI（Non Maskable Interrupt）と呼ばれるものです。X1では，リセット・スイッチの検出にNMIを使っており，このリセットを押すとどんな場合でも0066H番地をサブルーチン・コールします。そのため，ユーザー作成のマシン語プログラムが暴走しても，リセットを押せば戻ることができます(もっとも，0066H番地あたりのプログムまで壊れていれば不可能です)。

禁止・解除が可能な割り込みには3種類ありますが，X1はこのなかでもっとも機能の高いモード2割り込みを使っています。モード2の割り込みはZ80のIレジスタ（インタラプト・レジスタ）に割り込みが起こったときのコール先が入ったテーブルの先頭アドレスの上位バイトをセットします。下位アドレスは，周辺LSIからのベクター値ですが，X1では80C49にE4Hのコードの次に送る1バイトで決まります。
では実際に，キー入力割り込みの設定処理ルーチンをつくってみましょう。

```
        DI               ；割り込み禁止
        IM    2          ；モード2設定
        LD    HL, KEYIN
        LD    (KEYVCT), HL
        LD    A, H
        LD    I, A       ；番地の上位を設定
        LI    A, 0E4H    ；キーベクター設定コマンド
        CALL  TRANS49
        LD    A,L
        CALL  TRANS49    ；番地の下位を設定
        EI               ；割り込み解除
        RET

KEYVCT:DEEW KEYIN        ；キーベクターテーブル
          ⋮
KEYIN : PUSH AF          ；┐ 割り込み処理ルーチン
        PUSH HL          ；│ では，すべてのレジス
          ⋮              ；┘ タを保存する

        CALL  RECV49     ；┐
        LD    H, A       ；│ 2バイトのキーデータ
        CALL  RECV49     ；│ を取り込む
        LD    L, A       ；┘
          ⋮              ；割り込み処理本体

        POP   HL
        POP   AF
        EI               ；割り込み解除
        RETI
```

このようなプログラムを実行した後では，キー入力があるたびにKEYINがサブルーチン・コールされます。割り込みはいつかかるかわからないので，KEYINの最初では全レジスタを保存する必要があります（もっとも，自分でつくったプログラムのなかでは絶対にIX，IYは使わないというのであればこれらは保存する必要はない）。

Z80は割り込みがかかると次の割り込みが入らないように割り込み禁止にします（内部で自動的にDI命令を実行する）。そのため，KEYINの最後にEI命令を実行しなければなりません。また，RETIはサブルーチンから戻るだけでなく，外部に割り込み処理が終わったことを知らせるものです。

IOCSでは割り込みのテーブルが0052H，0053H番地にあるので，ここを書き換えても同様です。この場合，次のようになります。

```
        DI
        LD      HL, KEYIN
        LD      (0052H), HL
        EI
```

80C49からのキーデータは2バイトで，この構成は**図4-3**のようになっています。2バイト目はASCIIコードです。

注意しなければならない点は80C49に送る割り込みテーブルの下位は00であってはならず，最下位ビットは0（偶数）でなくてはならないということです。もし，00だと，それ以降割り込みがかからなくなり，キー入力を調べたいときは，送信要求コードのE6Hを送ってキーバッファを読み込まなければなりません。

図4 3 キーデータ(1バイト目)のビット構成

| | 7 (MSB) | 6 | 5 | 4 | 3 | 2 | 1 | 0 (LSB) |
|---|---|---|---|---|---|---|---|---|
| | ファンクション | キーデータが 有効 無効 | リピート | GRAPH | CAPS | カナ | SHIFT | CTRL |
| 0 | ●テンキー<br>●ファンクションキー<br>●TV キー<br>●カセット・キー | ●データ・コード(8ビット)が有効である<br>ヌルコード"00"以外が送られてきたとき | ●リピート・データである | ●GRAPHキーが押されている | ●CAPS キーが押されている<br>(LOCK されている) | ●カナキーが押されている<br>(LOCK されている) | ●SHIFT キーが押されている | ●CTRL キーが押されている |
| 1 | ●上記以外 | ●データコード(8ビット)が無効である<br>ヌル・コード"00"が送られてきたとき | ●1回目のデータである | ●GRAPH キーが離されている | ●CAPS キーが離されている | ●カナキーが離されている | ●SHIFT キーが離されている | ●CTRL キーが離されている |

# 4-3-2 タイマー,テレビのコントロール

**●テレビ・コントロール**

テレビのコントロールは次のように大きく2つに分けられます。

❶画面モード

テレビ画面,コンピュータ画面,スーパーインポーズでのコントラストなどを設定します。送信要求コードはE7Hで,後読バイト数は1～5です(**表4-3**)。

❷各種テレビ・コントロール

テレビのボリューム,チャンネルなどを設定します。送信要求コードはE7Hで,後続バイト数は1です(**表4-4**)。

表4 3 画面モード用コマンド

| 画面モード ＼ 送信コード(バイト数) | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|
| テレビ画面 | E7 | 05 | | | | |
| コンピュータ画面 | E7 | 05 | E7 | 08 | | |
| スーパーインポーズ 1<br>(コントラスト・ダウン) | E7 | 05 | E7 | 0F | E7 | 0A |
| スーパーインポーズ 2<br>(コントラスト・ノーマル) | E7 | 05 | E7 | 0F | | |

**表4-4 テレビ・コントロール用コマンド**

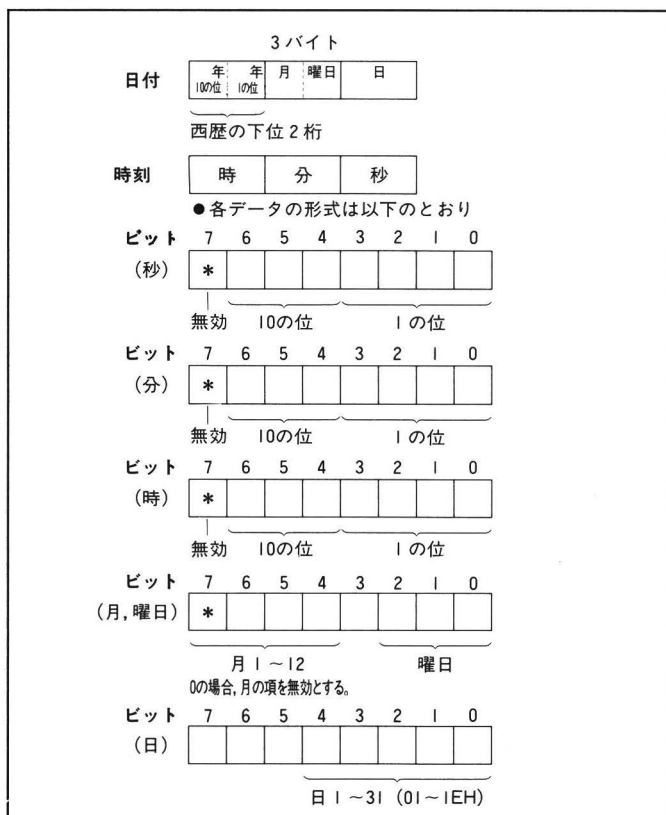
| 内容 ＼ 送信コード(バイト数) | 1 | 2 |
|---|---|---|
| ボリューム・アップ | E7 | 01 |
| ボリューム・ダウン | E7 | 02 |
| ボリューム・ノーマル（$\frac{42}{64}$階調） | E7 | 03 |
| 音声ミュート | E7 | 06 |
| チャンネル・アップ | E7 | 0B |
| チャンネル・ダウン | E7 | 0C |
| パワーオフ | E7 | 0D |
| パワーオン／オフ（トグル動作） | E7 | 0E |
| チャンネル1 〜 チャンネル12 | E7 | 10 〜 1B |
| パワーオン | E7 | 80 |

## ●タイマー

X1ではタイマー用ICとして，μPD1990を使っています。**図4-1**で示したようにこのICは80C49に管理されているので，タイマーの設定も80C49にコマンドを送って行います。日付，時刻は独立に設定・読み出しが可能です（**表4-2**）。後続バイト数は3で，設定と読み出しデータ・フォーマットは同じです（**図4-4**）。例として'85年2月28日金曜日12時25分45秒に設定してみましょう。

```
        LD    B, 8         ;送信バイト数
        LD    HL, DATA     ;送信データ先頭アドレス
LOOP:   LD    A, (HL)
        CALL  TRANS49
        INC   HL
        DJNZ  LOOP
        RET
```

```
;
DATA: DEFB 0ECH        ;日付設定送信コード
      DEFB 85H         ;'85年
      DEFB 25H         ; 2月, 金曜日
      DEFB 23H         ;23日
      ;
      DEFB 0EEH        ;時刻設定送信コード
      DEFB 12H         ;12時
      DEFB 25H         ;25分
      DEFB 45H         ;45秒
```

図4-4 タイマーのデータ・フォーマット

### ●テレビ・タイマー

X1はテレビ・タイマーの機能を持っており，スイッチのON/OFFやチャンネルなどが設定できます。HuBASICのASK命令では，1～7までの7個となっていますが，実際は0を加えて計8個のテレビ・タイマーを設定できます(**表4-5**)。

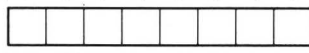
テレビ・タイマーをコントロールするための後続バイト数は6で，データ・フォーマットは**図4-5**に示すとおりです。なお，分，時，月，曜日，日はタイマーのデータ・フォーマットと同じなので省略しています。

なお,80C49は上記の他にカセットのコントロールも行っていますが，これについては4章で述べます。

**表4-5 テレビ・タイマー用コマンド**

| タイマー番号 | 設定コード | 読み出しコード |
|---|---|---|
| 0 | D0 | D8 |
| 1 | D1 | D9 |
| 2 | D2 | DA |
| 3 | D3 | DB |
| 4 | D4 | DC |
| 5 | D5 | DD |
| 6 | D6 | DE |
| 7 | D7 | DF |

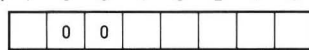
図4 5 テレビ・タイマーのデータ・フォーマット

●テレビ・タイマーのデータ構成

●インターバル



●ビット7が0，ビット6が1のときのみ，タイマー有効。

●あるタイマーが動作してから，指定時間（1～59分）経過後，再び同じ動作を行うタイマーをインターバル・タイマーという。

●コントロール内容

0：0～4ビットのコードを送る。
1：テレビがパワーオンしたのち，0～4ビットのコードを送る。

| データコード | コントロール内容 | データコード | コントロール内容 |
|---|---|---|---|
| 00 | タイマー動作 OFF | 80 | パワーオン |
| 01 | ボリューム アップ | 81 | パワーオン→ボリュームアップ |
| 02 | ボリューム ダウン | 82 | パワーオン→ボリュームダウン |
| 03 | ボリューム ノーマル | 83 | パワーオン→ボリュームノーマル |
| 06 | 音声ミュート | 86 | パワーオン→音声ミュート |
| 0B | チャンネル アップ | 8B | パワーオン→チャンネルアップ |
| 0C | チャンネル ダウン | 8C | パワーオン→チャンネルダウン |
| 0D | パワー オフ | | |
| 0E | パワーオン/オフ（トグル動作） | | |
| 10 | チャンネル1 | 90 | パワーオン→チャンネル1 |
| ～ | ～ | ～ | ～ |
| 1B | チャンネル12 | 9B | パワーオン→チャンネル12 |

# 第5章

# PSG



X1 では,音を発生させる回路に PSG (AY-3-8910) を使っており, この LSI に内蔵されている16個のレジスタに値を設定することにより, さまざまな音を発生することができます。また, この PSG は2つの汎用入出力ポートを持ち, 外部機器とデータの受け渡しが可能です。

本章では, この PSG の基本的な解説を中心に音楽演奏, ジョイスティックの入出力について述べていきます。

# 5-1 PSGのハードウェア

AY-3-8910(以下PSGと略)は，米国General Instrument (GI) 社によって開発された40ピンのLSIです。

PSGは，同じGI社のCPU（CPI600など）との接続をもとに設計されていますが，X1のように80系のCPUとの接続も可能です。

ただし，PSGの反応速度はあまり速くなく，そのままつないだのでは，連続してデータを書き込んだときに取りこぼしてしまうことがあります。

このため，X1の回路では，PSGのアドレス・レジスタをアクセスするときに，CPUに対して自動的に１つ分の待ちサイクルを入れることによって，これを解決しています。

PSGのピン配置を**図5-1**に，Z80とのインターフェイスを**図5-2**に示します。PSGに関するI/Oポートは２つで，I/Oアドレスはそれぞれ1B＊＊Hと1C＊＊H（下位８ビットは無効で，どんな値でもよい）です。

**図5-1 PSGのピン配置図**

AY-3-8910

| 信号 | ピン | ピン | 信号 |
|---|---|---|---|
| Vss(GND) | 1 | 40 | Vcc(+5V) |
| N.C. | 2 | 39 | TEST 1 |
| ANALOG CH.B | 3 | 38 | ANALOG CH.C |
| ANALOG CH.A | 4 | 37 | DA0 |
| N.C. | 5 | 36 | 〃 1 |
| IOB7 | 6 | 35 | 〃 2 |
| 〃 6 | 7 | 34 | 〃 3 |
| 〃 5 | 8 | 33 | 〃 4 |
| 〃 4 | 9 | 32 | 〃 5 |
| 〃 3 | 10 | 31 | 〃 6 |
| 〃 2 | 11 | 30 | 〃 7 |
| 〃 1 | 12 | 29 | BC1 |
| 〃 0 | 13 | 28 | BC2 |
| IOA7 | 14 | 27 | BDIR |
| 〃 6 | 15 | 26 | TEST 2 |
| 〃 5 | 16 | 25 | A8 |
| 〃 4 | 17 | 24 | A9 |
| 〃 3 | 18 | 23 | $\overline{RESET}$ |
| 〃 2 | 19 | 22 | CLOCK |
| 〃 1 | 20 | 21 | IOA0 |

| 記号 | 意味 |
|---|---|
| Vss | GND |
| Vcc | 5V |
| N.C. | 未使用 |
| CH.A | チャンネルA音声出力 |
| CH.B | チャンネルB音声出力 |
| CH.C | チャンネルC音声出力 |
| IOA0～IOA7 | ポートA入出力 |
| IOB0～IOB7 | ポートB入出力 |
| DA0～DA7 | データ・バス |
| BC1,BC2 | バス・コントロール |
| BDIR | バスの方向 |
| CLOCK | クロック入力(X1の場合2MHz) |
| A8, A9 | チップ・セレクト |
| TEST1,TEST2 | メーカーのテスト端子,接続しない |
| $\overline{RESET}$ | リセット |

この1B＊＊H, 1C＊＊Hと読み出しか，書き込みかを示す$\overline{WR}$，$\overline{RD}$により，PSGのBDIR，BC1をデコードしています。BDIR，BC1はPSGの2つのポートをセレクトするのに使います（**表5-1**)。

たとえば，1B＊＊Hと$\overline{WR}$がアクティブのとき，Z80は，PSGのデータ・ポートに書き込んでいることになります。

PSGは，外部より与えられたクロックを分周して音を発生します。この入力するクロックによって，その回路での発生する音域が決められます。

X1では，このクロックに2 MHz，ちょうどCPUのクロックを半分にしたものを使っています。

**5-2**で述べる音の高さを決める式から，X1で発生させることのできる音域は，125 KHzから30.5 Hzです。

**図5-2 PSGとZ80とのインターフェイス**

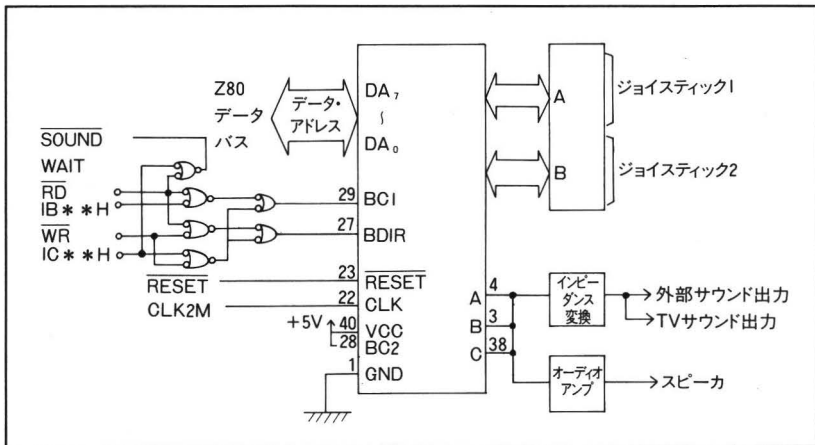



**表5-1 BC1，BC2，BDIR**

| I/Oアドレス | BDIR | BC1 | 機　能 | I/O |
|---|---|---|---|---|
| ―――― | 0 | 0 | ノンアクティブ | ―― |
| 1B＊＊H | 0 | 1 | PSGから読み出し | IN |
| | 1 | 0 | PSGへの書き込み | OUT |
| 1C＊＊H | 1 | 1 | アドレスをラッチする | OUT |

＊BC2は常に1

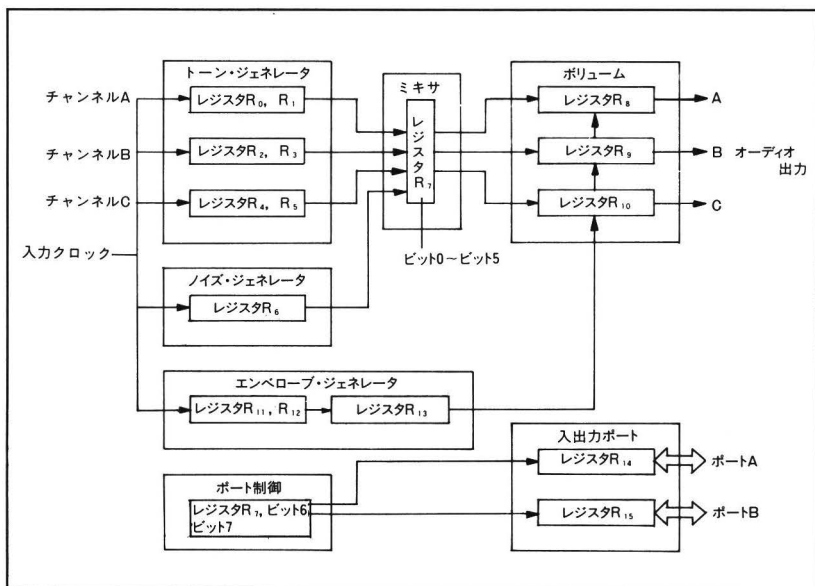
# 5-2 PSGのレジスタ

PSGは内部に16個（このうち音の発生用には14個）の８ビット・レジスタを持ち，これらにデータを書き込むことで音を発生することができます。PSGのブロック・ダイアグラムを**図5-3**に，レジスタを**表5-2**に示します。

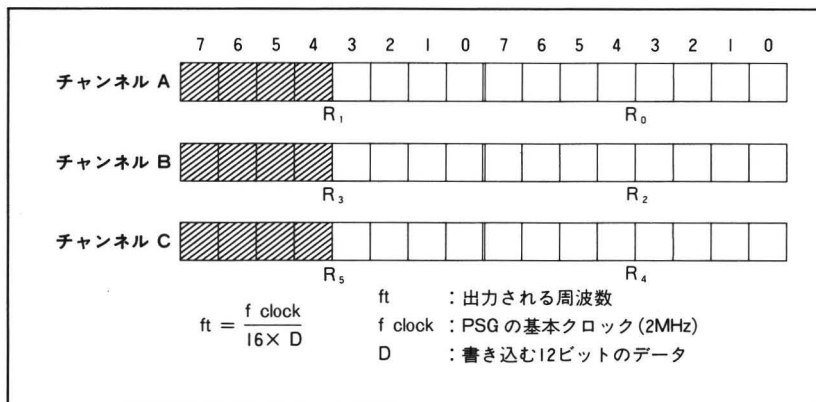
❶ $\mathbf{R_0}$～$\mathbf{R_5}$（周波数を決める）

$R_0$～$R_5$はトーン・レジスタと呼ばれ，音の高さを決めるものです。これらは，それぞれ$R_0$と$R_1$，$R_2$と$R_3$，$R_4$と$R_5$というように２つ１組で12ビット・レジスタとして扱います（**図5-4**）。$R_1$，$R_3$，$R_5$の上位４ビットは無効です。

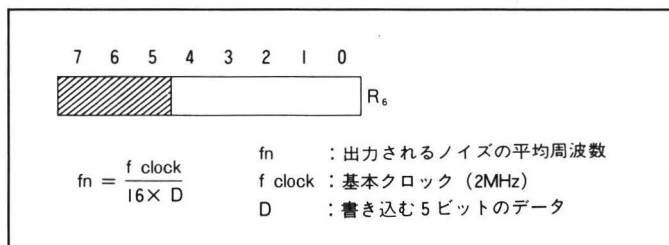
12ビットですから設定できる値は０～4095で**図5-4**の計算式から，設定する値が１のとき125 KHz，4095のとき30.5 Hzとなります。

**図5-3** PSGのブロック・ダイアグラム

## 表5-2 PSGのレジスタ

| レジスタ | 機能 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
|---|---|---|---|---|---|---|---|---|---|---|
| $R_0$ | チャンネル A の周波数 | 下位8ビット・データ | | | | | | | | |
| $R_1$ | | | | | | 上位4ビット・データ | | | | |
| $R_2$ | チャンネル B の周波数 | 下位8ビット・データ | | | | | | | | |
| $R_3$ | | | | | | 上位4ビット・データ | | | | |
| $R_4$ | チャンネル C の周波数 | 下位8ビット・データ | | | | | | | | |
| $R_5$ | | | | | | 上位4ビット・データ | | | | |
| $R_6$ | ノイズの平均周波数 | | | | 5ビット・データ | | | | | |
| $R_7$ | ミキシング,ポート制御 | 入出力選択 | | ノイズ | | | トーン | | | 対応するビットが1のときオフ,0のときオン |
| | | ポートB | ポートA | C | B | A | C | B | A | |
| $R_8$ | チャンネル A の音量 | | | | M | 4ビット・データ | | | | M = 0のとき下位4ビット・データによる音量調節<br>M = 1のとき<br>エンベロープ作動 |
| $R_9$ | チャンネル B の音量 | | | | M | 4ビット・データ | | | | |
| $R_{10}$ | チャンネル C の音量 | | | | M | 4ビット・データ | | | | |
| $R_{11}$ | エンベロープ周期 | 下位8ビット・データ | | | | | | | | |
| $R_{12}$ | | 上位8ビット・データ | | | | | | | | |
| $R_{13}$ | エンベロープ波形 | | | | | CONT | ATT | ALT | HOLD | |
| $R_{14}$ | ポート A | | | | | | | | | $R_7$ の入出力選択が1のとき出力,0のとき入力 |
| $R_{15}$ | ポート B | | | | | | | | | |

## 図5-4 トーン・コントロール

❷ $R_6$（ノイズ）

$R_6$ はノイズの平均周波数を決めるものです。ノイズは発生回路を１つしか持っていません。**図5-5**に示すように下位５ビットが有効で，設定できる値は0～31となります。**図5-5**の計算式からノイズの平均周波数は，設定する値が１のとき125 KHz，31のとき４ KHzです。

**図5-5 ノイズ・コントロール**

❸ $R_7$（ミキシング，入出力制御）

$R_7$ は，トーンとノイズの出力チャンネルを決める他に，$R_{14}$，$R_{15}$ の２つの入出力ポートの入出力の方向を決めます。$R_7$ のビット構成を**図5-6**に示します。

❹ $R_8$～$R_{10}$（音量）

$R_8$～$R_{10}$ は，それぞれのチャンネルに対してボリュームの働きをします。

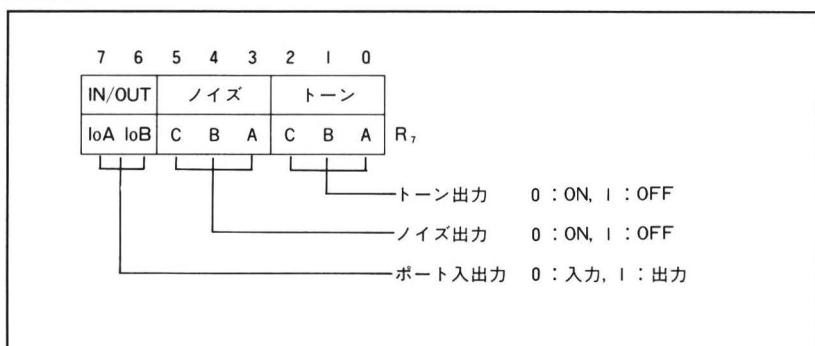
下位４ビットで表される０～15の値で，０のとき無音，15のとき最大となります。ただし，この音量の値が意味を持つのは，ビット４が０（エンベロープOFF）のときです（**図5-7**）。

❺ $R_{11}$，$R_{12}$（エンベロープ周期）

$R_{11}$ と $R_{12}$ を組み合わせて16ビットとして使い，エンベロープの周期（**図5-9**の $t_E$）を決めます（**図5-8**）。

❻ $R_{13}$（エンベロープ波形）

エンベロープの形状を決めます。下位４ビットが有効で，形状は**図5-9**に示すものがあります。

図5 6 $R_7$のビット構成

<table>
<tr><th>ビット</th><th>説　明</th></tr>
<tr><td>$B_2$〜$B_0$</td><td>出力させたいトーン・チャンネルの選択
<table>
<tr><th>$B_2$</th><th>$B_1$</th><th>$B_0$</th><th colspan="3">トーン出力チャンネル</th></tr>
<tr><td>0</td><td>0</td><td>0</td><td>C</td><td>B</td><td>A</td></tr>
<tr><td>0</td><td>0</td><td>1</td><td>C</td><td>B</td><td>—</td></tr>
<tr><td>0</td><td>1</td><td>0</td><td>C</td><td>—</td><td>A</td></tr>
<tr><td>0</td><td>1</td><td>1</td><td>C</td><td>—</td><td>—</td></tr>
<tr><td>1</td><td>0</td><td>0</td><td>—</td><td>B</td><td>A</td></tr>
<tr><td>1</td><td>0</td><td>1</td><td>—</td><td>B</td><td>—</td></tr>
<tr><td>1</td><td>1</td><td>0</td><td>—</td><td>—</td><td>A</td></tr>
<tr><td>1</td><td>1</td><td>1</td><td>—</td><td>—</td><td>—</td></tr>
</table></td></tr>
<tr><td>$B_5$〜$B_3$</td><td>出力させたいノイズ・チャンネルの選択
<table>
<tr><th>$B_5$</th><th>$B_4$</th><th>$B_3$</th><th colspan="3">ノイズ出力チャンネル</th></tr>
<tr><td>0</td><td>0</td><td>0</td><td>C</td><td>B</td><td>A</td></tr>
<tr><td>0</td><td>0</td><td>1</td><td>C</td><td>B</td><td>—</td></tr>
<tr><td>0</td><td>1</td><td>0</td><td>C</td><td>—</td><td>A</td></tr>
<tr><td>0</td><td>1</td><td>1</td><td>C</td><td>—</td><td>—</td></tr>
<tr><td>1</td><td>0</td><td>0</td><td>—</td><td>B</td><td>A</td></tr>
<tr><td>1</td><td>0</td><td>1</td><td>—</td><td>B</td><td>—</td></tr>
<tr><td>1</td><td>1</td><td>0</td><td>—</td><td>—</td><td>A</td></tr>
<tr><td>1</td><td>1</td><td>1</td><td>—</td><td>—</td><td>—</td></tr>
</table></td></tr>
<tr><td>$B_7$, $B_6$</td><td>I/O ポート(ジョイスティック)入出力方向指定
<table>
<tr><th>$B_7$</th><th>$B_6$</th><th>B<br>JS 2</th><th>A<br>JS 1</th><th>入出力制御</th></tr>
<tr><td>0</td><td>0</td><td>IN</td><td>IN</td><td>ジョイスティック(JS)<br>1, 2共に入力</td></tr>
<tr><td>0</td><td>1</td><td>IN</td><td>OUT</td><td>JS 2 ：入力<br>JS 1 ：出力</td></tr>
<tr><td>1</td><td>0</td><td>OUT</td><td>IN</td><td>JS 2 ：出力<br>JS 1 ：入力</td></tr>
<tr><td>1</td><td>1</td><td>OUT</td><td>OUT</td><td>JS 1, 2共に出力</td></tr>
</table></td></tr>
</table>

### ❼ $R_{14}$，$R_{15}$（入出力ポート）

$R_{14}$と$R_{15}$はそれぞれ8ビットの入出力ポートです。入出力の決定は，$R_7$のビット6と7で行います。

入力ポートとして用いた場合，Z80にはこのレジスタがアクセスされた時点の内容を返します。

出力ポートとして用いた場合，PSGは設定された値をそのまま出力します。

図5 7 音量設定

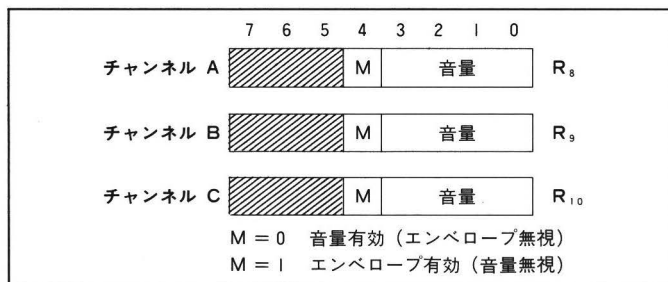



図5 8 エンベロープ周期設定

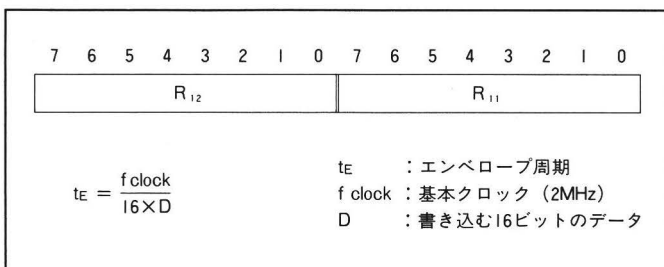



図5-9 エンベロープ波形

| レジスタ $R_{13}$ の下位4ビット | 16進値 | エンベロープ・パターン |
|---|---|---|
| 0 0 — — | 0 ~ 3 | |
| 0 1 — — | 4 ~ 7 | |
| 1 0 0 0 | 8 | |
| 1 0 0 1 | 9 | |
| 1 0 1 0 | A | |
| 1 0 1 1 | B | |
| 1 1 0 0 | C | |
| 1 1 0 1 | D | |
| 1 1 1 0 | E | |
| 1 1 1 1 | F | |

$t_E$

# 5-3 音階データ

PSGで音楽演奏をする場合，BASICではPLAY文を使って音階や長さを指定して行いますが，マシン語では**5-2**で示した式からPSGに与えるデータをまず求めなければなりません。**表5-3**は，音階とそれに対応する周波数およびトーン・ジェネレータ用に計算したデータです。この表があれば，後は音の長さを測りながらトーン・ジェネレータにデータを書き込むことで演奏ができます。

いま，ひとつの例としてチャンネルAから時報や楽器のチューニングに使われる4オクターブ目のA（ラ）の音を出すことを考えてみましょう。**表5-3**からトーン・ジェネレータに設定するデータは011Eであることがわかるので，この音を出す順序は次のようになります。

①チャンネルAのトーン出力をONにする。

$R_7$に3E（トーン出力AのみON）を書き込む。

②音量をセットする。

$R_8$に0～15のデータを書き込む。

③トーン・ジェネレータにデータを書き込む。

データは011Eなので，$R_0$に1E，$R_1$に01を書き込む。

以上のことは，BASICのSOUND文で簡単に試すことができます。

マシン語でPSGにデータを設定するには，

①I/Oポートの1C＊＊H番地に，書き込みたいPSGのレジスタ番号を出力する。

②I/Oポートの1B＊＊H番地に，データを出力する。

という手順で行います。したがってSOUND文に相当するサブルーチンは，Aレジスタにレジスタ番号，Eレジスタにデータが入っているものとして次のように書けます。

```
        LD      B, 1CH
```

```
OUT  (C), A         ;レジスタ番号を出力
DEC  B
OUT  (C), E         ;データを出力
RET
```

**表5-3 音階とトーン・ジェネレータ用のデータ**

| ON | K. | Tone(Hz) | Dec. | Hex. | ON | K. | Tone(Hz) | Dec. | Hex. |
|---|---|---|---|---|---|---|---|---|---|
| C | 1 | 32.703 | 3824 | 0EF0 | C | 5 | 523.248 | 241 | 00F1 |
| C# | 1 | 34.648 | 3610 | 0E1A | C# | 5 | 554.368 | 227 | 00E3 |
| D | 1 | 36.708 | 3407 | 0D4F | D | 5 | 587.328 | 215 | 00D7 |
| D# | 1 | 38.891 | 3216 | 0C90 | D# | 5 | 622.256 | 203 | 00CB |
| E | 1 | 41.203 | 3036 | 0BDC | E | 5 | 659.248 | 192 | 00C0 |
| F | 1 | 43.654 | 2865 | 0B31 | F | 5 | 698.464 | 181 | 00B5 |
| F# | 1 | 46.249 | 2705 | 0A91 | F# | 5 | 739.984 | 171 | 00AB |
| G | 1 | 48.999 | 2553 | 09F9 | G | 5 | 783.984 | 161 | 00A1 |
| G# | 1 | 51.913 | 2410 | 096A | G# | 5 | 830.608 | 152 | 0098 |
| A | 1 | 55.000 | 2275 | 08E3 | A | 5 | 880.000 | 144 | 0090 |
| A# | 1 | 58.270 | 2147 | 0863 | A# | 5 | 932.320 | 136 | 0088 |
| B | 1 | 61.735 | 2027 | 07EB | B | 5 | 987.760 | 129 | 0081 |
| C | 2 | 65.406 | 1913 | 0779 | C | 6 | 1046.496 | 121 | 0079 |
| C# | 2 | 69.296 | 1806 | 070E | C# | 6 | 1108.736 | 115 | 0073 |
| D | 2 | 73.416 | 1705 | 06A9 | D | 6 | 1174.656 | 108 | 006C |
| D# | 2 | 77.782 | 1609 | 0649 | D# | 6 | 1244.512 | 102 | 0066 |
| E | 2 | 82.406 | 1519 | 05EF | E | 6 | 1318.496 | 97 | 0061 |
| F | 2 | 87.308 | 1434 | 059A | F | 6 | 1396.928 | 91 | 005B |
| F# | 2 | 92.498 | 1353 | 0549 | F# | 6 | 1479.968 | 86 | 0056 |
| G | 2 | 97.998 | 1278 | 04FE | G | 6 | 1567.968 | 82 | 0052 |
| G# | 2 | 103.826 | 1206 | 04B6 | G# | 6 | 1661.216 | 77 | 004D |
| A | 2 | 110.000 | 1138 | 0472 | A | 6 | 1760.000 | 73 | 0049 |
| A# | 2 | 116.540 | 1075 | 0433 | A# | 6 | 1864.640 | 69 | 0045 |
| B | 2 | 123.470 | 1014 | 03F6 | B | 6 | 1975.520 | 65 | 0041 |
| C | 3 | 130.812 | 958 | 03BE | C | 7 | 2092.992 | 62 | 003E |
| C# | 3 | 138.592 | 904 | 0388 | C# | 7 | 2217.472 | 58 | 003A |
| D | 3 | 146.832 | 853 | 0355 | D | 7 | 2349.312 | 55 | 0037 |
| D# | 3 | 155.564 | 806 | 0326 | D# | 7 | 2489.024 | 52 | 0034 |
| E | 3 | 164.812 | 760 | 02F8 | E | 7 | 2636.992 | 49 | 0031 |
| F | 3 | 174.616 | 718 | 02CE | F | 7 | 2793.856 | 47 | 002F |
| F# | 3 | 184.996 | 678 | 02A6 | F# | 7 | 2959.936 | 44 | 002C |
| G | 3 | 195.996 | 640 | 0280 | G | 7 | 3135.936 | 42 | 002A |
| G# | 3 | 207.652 | 604 | 025C | G# | 7 | 3322.432 | 40 | 0028 |
| A | 3 | 220.000 | 570 | 023A | A | 7 | 3520.000 | 38 | 0026 |
| A# | 3 | 233.080 | 538 | 021A | A# | 7 | 3729.280 | 36 | 0024 |
| B | 3 | 246.940 | 508 | 01FC | B | 7 | 3951.040 | 34 | 0022 |
| C | 4 | 261.624 | 480 | 01E0 | C | 8 | 4185.984 | 32 | 0020 |
| C# | 4 | 277.184 | 453 | 01C5 | C# | 8 | 4434.944 | 30 | 001E |
| D | 4 | 293.664 | 428 | 01AC | D | 8 | 4698.624 | 29 | 001D |
| D# | 4 | 311.128 | 404 | 0194 | D# | 8 | 4978.048 | 27 | 001B |
| E | 4 | 329.624 | 381 | 017D | E | 8 | 5273.984 | 26 | 001A |
| F | 4 | 349.232 | 360 | 0168 | F | 8 | 5587.712 | 24 | 0018 |
| F# | 4 | 369.992 | 340 | 0154 | F# | 8 | 5919.872 | 23 | 0017 |
| G | 4 | 391.992 | 321 | 0141 | G | 8 | 6271.872 | 22 | 0016 |
| G# | 4 | 415.304 | 303 | 012F | G# | 8 | 6644.864 | 21 | 0015 |
| A | 4 | 440.000 | 286 | 011E | A | 8 | 7040.000 | 20 | 0014 |
| A# | 4 | 466.160 | 270 | 010E | A# | 8 | 7458.560 | 19 | 0013 |
| B | 4 | 493.880 | 255 | 00FF | B | 8 | 7902.080 | 18 | 0012 |

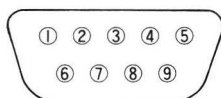
# 5-4 ジョイティック

X1にはジョイスティック端子が2つ出ています。この端子はアタリ社仕様に準拠しているので，アタリ社仕様のジョイスティックであれば使用できます。日本で売られているパソコンでは，PC-6001, MSXなどもアタリ仕様なので，それらのジョイスティックを流用できます。

ジョイスティック端子は**図5-10**のような信号がでています。

**図5-10 ジョイスティックのピンの配置図**

**X1側 ジョイスティック1, 2**

①②③④⑤
⑥⑦⑧⑨

AMP 9ピン相当

| 端子番号 | X1 信号名 | | アタリ社信号名 |
|---|---|---|---|
| | ジョイスティック1 | ジョイスティック2 | |
| 1 | IOA0 | IOB0 | FWD （入力） |
| 2 | IOA1 | IOB1 | BACK （入力） |
| 3 | IOA2 | IOB2 | LEFT （入力） |
| 4 | IOA3 | IOB3 | RIGHT （入力） |
| 5 | IOA4 | IOB4 | +5 V |
| 6 | IOA5 | IOB5 | トリガーボタン1（入力） |
| 7 | IOA6 | IOB6 | トリガーボタン2（入力） |
| 8 | GND | GND | 出力 |
| 9 | IOA7 | IOB7 | GND |

HuBASICではトリガーボタンは1つだけしかサポートしていませんが，ハード的にはトリガーボタン2つまで判別する能力を持っています。

また，本来この端子は汎用の入出力I/Oポートであり，X1では1～7番ピンと9番ピン（IOA0～7，IOB0～7）すべてが入力・出力どちらにも使用できます。ですから，ジョイスティック以外にも広く使用できそうです。

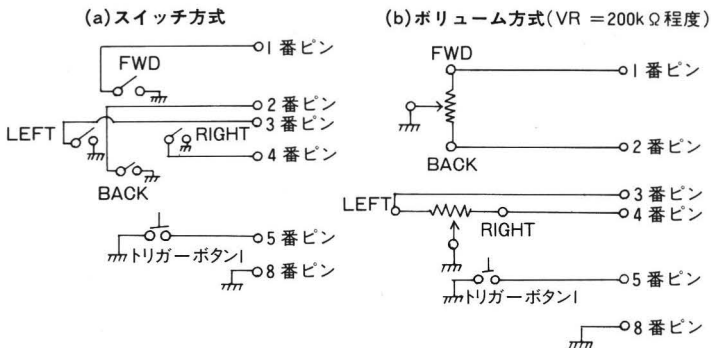
**図5-11**にジョイスティックの接続方法を示します。自作するときに使ってください。

ジョイスティックを使うには，PSG の $R_7$ に 2 つのポートの入出力の方向を設定しなければなりません。ビット 7， 6 がそれぞれポート B，A に対応しています。0 のときが入力で 1 のときが出力となります。

通常ジョイスティックは入力モードとして使うので，ビット 7， 6 は 0 にします。下位 6 ビットはサウンドの方で使っているのでまちがって変えてしまわないよう気をつけてください。

ポートの入出力方向を決めたら，後は $R_{14}$，$R_{15}$ に対して入出力を行います。出力は，**5-3**で解説したサブルーチンでできます。入力は，たとえば $R_{14}$ から A レジスタに入力するサブルーチンは次のようになります。

```
LD    A, 14    ; R14
LD    B, 1CH
OUT   (C), A   ; レジスタ番号を出力
DEC   B
IN    A, (C)   ; データ入力
RET
```

**図5-11 ジョイスティックの接続方法**

ジョイスティックより入力した値は，**図5-12**で示されるように，ビットごとに対応しています。

それぞれ0のときが，その方向または，スイッチが押されている場合です。

たとえば，

```
DBH (1101 1011)
```

とき，ジョイスティックは，左にかたむけられ，トリガー1が押されていることになります。

**図5-12 ジョイスティックからの入力データ**

ビット

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| ― | トリガー2 | トリガー1 | ― | RIGHT | LEFT | BACK | FWD |

# 第6章

## カセット



X1 のカセットテープデッキは，電磁メカ仕様です。ふたを閉めることはできませんが，そのほかの操作はすべてソフトウエアで可能です。

この章では，カセットテープデッキのコントロール方法から，その記録方式であるシャープ PWM について解説していきます。また，ツールとして MZ-2000/80B/1200/700/80K/C とテープの読み込み，書き込みができるコンバータとバックアップ・プログラムを紹介します。

# 6-1 カセットのコントロール

X1は，カセットデッキのコントロールを80C49が行っています。これは，MZやX1特有の豊富なカセット・コントロール機能をサポートするために設けられており，CPUの負担を軽くしています。ただし，X1Dはカセットを駆動する回路を持っておらず，APSSや早送りなどの機能はありません。

## 6-1-1 コントロール・コマンド

プログラムでカセットをコントロールする場合は，80C49に対してコマンドを送ります。これは4章で説明したように直接送るのではなく，8255を介して行います。

カセット・コントロール用コマンドは，**表6-1**に示すように2バイトであり，1バイト目はE9Hで続く1バイトが動作を示します。たとえば，巻戻しをする場合，E9H+04Hを送ればよいので，プログラムは次のようになります。

```
TRANS49  EQU   0B54H    ; PUT CODE TO 80C49
         LD    A, 0E9H
         CALL  TRANS49
         LD    A, 04H   ; REW    COMMAND
         CALL  TRANS49
```

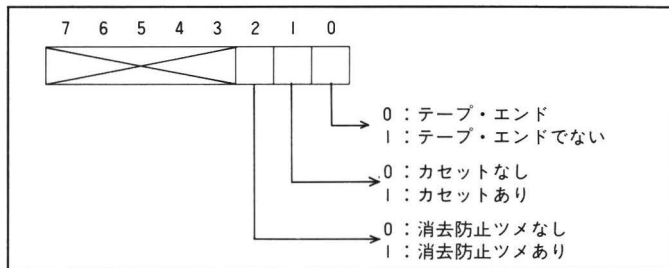
**表6-1 カセット・コントロール用コマンド**

| 動　作 | 送信コード |
|---|---|
| EJECT | E9, 00 |
| STOP | E9, 01 |
| PLAY | E9, 02 |
| FF | E9, 03 |
| REW | E9, 04 |
| APSS FF | E9, 05 |
| APSS REW | E9, 06 |
| REC | E9, 0A |

## 6-1-2 カセットのセンス

カセットが現在どのような動作をしているか，その状態を調べるのが，カセットの状態信号読みだしコマンドです。このコマンドは1バイトで，EBHです。

80C49は，このコマンドをうけると，そのときの状態信号を1バイトにして，Z80に返してきます。ただし，使われているのは下位3ビットのみでビット0から順に，テープ・エンド検出，テープ有無検出，ツメ検出にあてられています（**図6-1**）。この状態信号を使えば巻戻した後ふたをあけたり，頭出しをした後再生状態にするといったことができます。

図6-1 **カセットの状態コード**

## 6-1-3 カセットの動作状態

80C49は，現在のカセットメカの動作状態をZ80に送り返すことができます。返されるデータは，**表6-1**のコントロール用コマンドの2バイト目と同じです。このデータをZ80が読み込むことで現在カセットメカがどのような動作をしているのかを推測することができます。読み込むコマンドはEAHで，これを80C49に送ると80C49は，1バイトのコードを返してきます。

たとえば，早送りのコマンド（E9H，03H）を実行した後，このコマンドを送ると最後のコードである03Hが返ってきます。このプログラムは次のようになります。

```
TRANS49   EQU   0B54H
RECV49    EQU   0B49H    ;GET DATA FROM 80C49
                         ; DATA ->A reg.
          LD    A, 0EAH  ; GET COMMAND
          CALL  TRANS49
          CALL  RECV49
```

# 6-2 シャープPWM方式

シャープPWM方式は，MZシリーズ用に使われたテープへの記録方式ですが，信頼性が高いためX1でもこれが採用されています。

記録スピードは，2700ビット/秒（ボー）で，MZシリーズの1200ボー，2000ボーに比べても早くなっています。

2700ボーというと，1秒間に約377バイト分記録することになります。

PWM方式とは，Pulses Width Modulation(パルス幅変調)の略で，ビットの1と0をHからLまでの時間の違いを利用して記録するものです（**図6-2**）。0のときは250μSで1サイクル，1のときは500μSで書き込みます。たとえば，1001を書き込むと**図6-3**のようになります。この図からデータ0が多ければ，1が多い場合より高速にセーブできることがわかります。ですから2700ボーというのは正確な値ではなく，1と0が同じ個数のときの値です（より正確に言うときっかり2700ボーではなく，もうすこし低くなります。興味のある方は計算してみてください）。

図6-2 **PWM 方式**

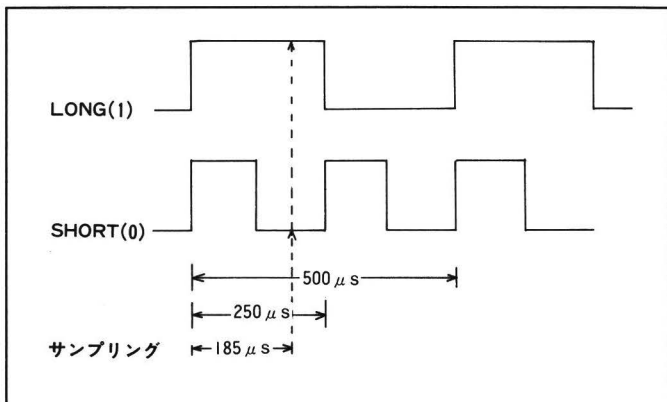
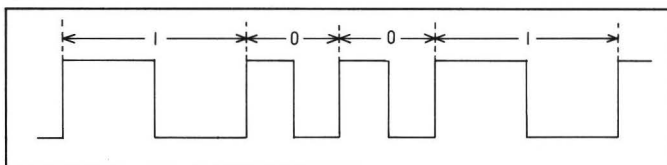



図6-3 1001の波形

逆に読み込むときは，波形がHになってから185$\mu$S後をサンプリングしても，もしこのときHならば1，Lならば0というように判断しています。

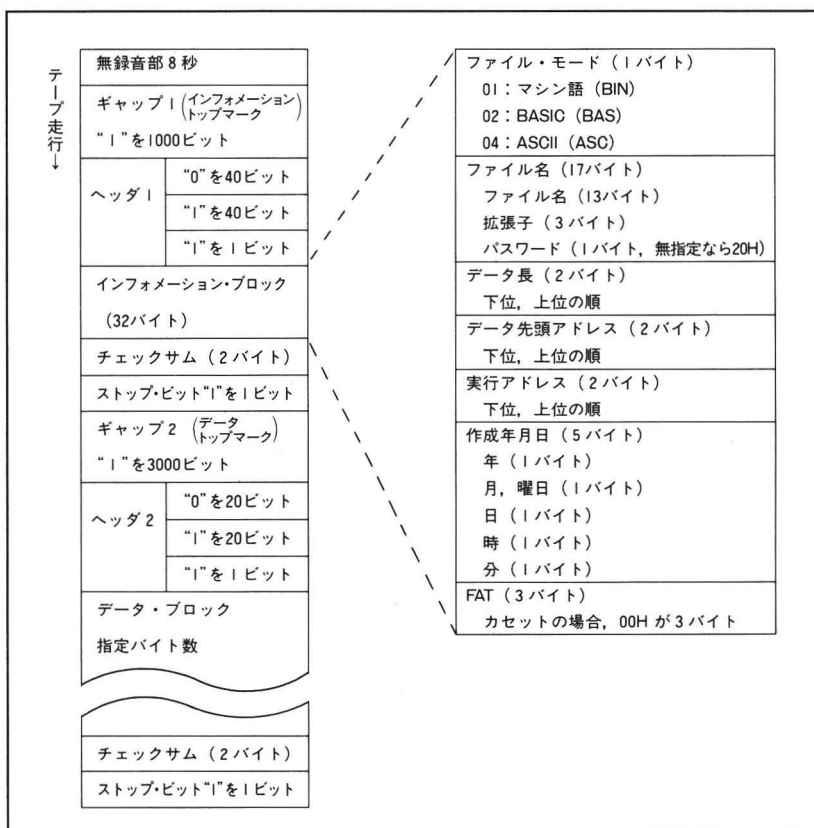
通常データのやり取りは，1バイト単位で行うので，書き込んだり，読み込んだりするには，上述したようなことを8回繰り返せばよいわけですが，実際には1ビット余計に書き込まれます。これがスタート・ビットで，必ず1です(**図6-4**)。

図6-4 **1バイトの構成**

| スタートビット "1" | MSB 7 | 6 | 5 | 4 | 3 | 2 | 1 | LSB 0 |
|---|---|---|---|---|---|---|---|---|

# 6-3 テープ・フォーマット

カセットテープに，プログラムを記録するときの形式は，**図6-5**に示すとおりです。テープへの記録は，大きく2つに分けられ，1つはインフォメーション・ブロック，もう1つはデータ・ブロックです。インフォメーション・ブロックは32バイトで構成され，ここにファイル名やプログラムの長さ，日付などが入ります。

図6-5 **テープ・フォーマット**

チェックサムは，読み込みのときにそのデータが間違っていないかどうかを検出するために設けられています。書き込みのときビット１の個数を数えていき，その数をチェックサムとして書き込みます。そして，読み込みのときも同様にビット１の個数を数えていきチェックサムと比較します。もしこの値が一致しないとチェックサム・エラーになります。

データ・ブロックは，ファイル・モードによって異なります。ファイル・モードは４種類（マシン語ファイル，BASICテキスト・ファイル，ASCIIセーブされたBASICテキスト・ファイル，データ・ファイル）ありますが，セーブされる形式は２種類です。

マシン語プログラムとBASICテキストのときが最も単純で，インフォメーション・ブロックで示されるデータの長さだけ書き込んだあと，２バイトのチェックサムを書き込むだけです。

ASCIIセーブやデータ・ファイルでは，ブロッキングされてセーブします。この場合，データは256バイトごとに分割して記録されます。ブロッキング・セーブの場合は，

| ギャップ２－>ヘッダ２－>データ |
|---|

を繰り返すことで65536ブロックまでのデータを記録することができます。

# 6-4 ボーレート,フォーマットの変換

**6-3**で述べたのは，HuBASICでのことであって，マシン語を使えばある程度自由にボーレートやフォーマットが独自に設定できます。ここでは，これを利用してMZ-2000/80B/1200/80K/Cとテープを使ってファイルのコンバートする方法とそのプログラムを紹介します。

これらのMZ系のマシンにもHuBASICが市販されており，なかでもバージョン2.0が最も普及していると思います。また，MZ-700にはX1と同様，SHARP HuBASICと称して本体に標準で添付されています。

これらのHuBASICは，コマンド体系が統一されており，中間言語も大部分が共通であるため，異なるマシンのBASICプログラムを転用することが比較的容易です。マシン語のプログラムもBASIC同様に読み込み，書き込みが可能です。マシン語の場合でもIOCSのエントリーアドレスが大部分共通なのでハードに依存したプログラムでない場合はそのまま走らせることができるでしょう。

各HuBASICのボーレートとフォーマットの違いを**図6-6**に示します。ボーレートはディレイ・カウンタD, S1, S2, L1, L2の値によって決定されます(**図6-7**)。時間まちのサブルーチンはIOCSの0DBFH～0DC6Hにあり，このプログラムは，

```
        LD      A,2EH       ;ディレイ・カウンタD
        NOP
LOOP:   DEC     A
        JP      NZ, LOOP
        RET
```

となっており，Aレジスタの回数分空ループしています。

インフォーメーション部はX1では32バイト(20H)で，MZ

シリーズのHuBASICではS-BASICと合わせるためか128バイト(80H)あります。しかし，通常使用されるのは先頭から32バイトまでであり，その構造はX1とまったく同じです。

トップマークは(ギャップやヘッダ)はその名称のとおりインフォメーション部やデータ部が始まる目印で，この構造がX1とMZシリーズではまったく反対であり，トップマークの長さも各HuBASICによってまちまちになっています(**図6-8**)。

以上の違いを変更すると他のHuBASICのテープファイルの読み込み，書き込みが可能になります。

コンバート・プログラムを**リスト6-1**に示します。実行するとロード・フォーマット，セーブ・フォーマットの順に聞いてくるので，1～3の数字で機種を答えてください。

インフォメーション部が128バイトあるときに，X1でそのまま読み込ませては，ワーク・エリアが足りずに暴走する心配があります。このプログラムではFE60H番地からにワーク・エリアを設け，そこから32バイトだけ1480H番地へ転送させて使用しています。1480H番地が本来インフォメーション部分が入るエリアです。

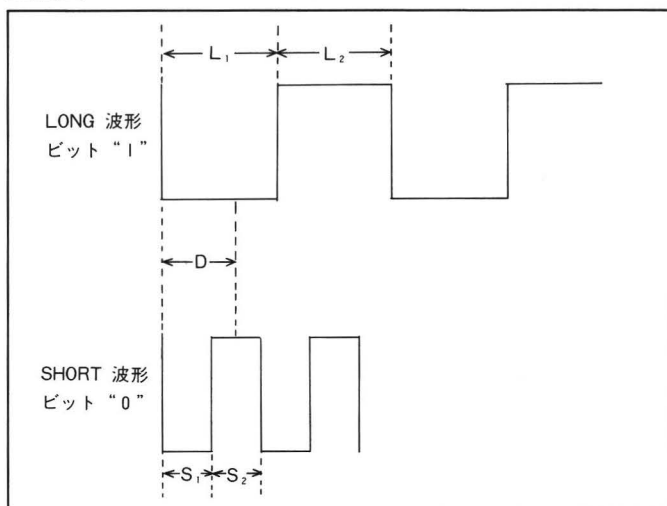
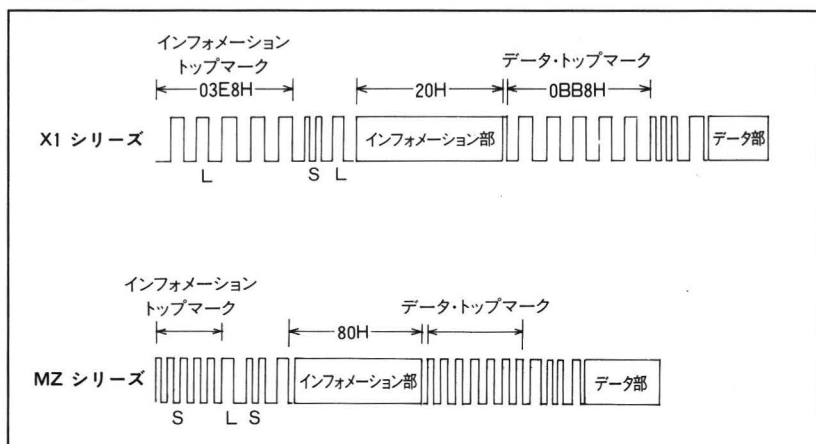
**図6-6 各 HuBASIC の違い**

| | ボーレート<br>(baud) | インフォメーション長<br>(バイト) | LOAD 時<br>ディレイカウンタ<br>D | LOAD 時<br>トップマーク<br>論理 | SAVE 時<br>ディレイ・カウンタ<br>SHORT Low<br>S1 | SAVE 時<br>ディレイ・カウンタ<br>SHORT Hi<br>S2 | SAVE 時<br>ディレイ・カウンタ<br>LONG Low<br>L1 | SAVE 時<br>ディレイ・カウンタ<br>LONG Hi<br>L2 | SAVE 時<br>インフォメーション<br>トップマーク長<br>(サイクル) | SAVE 時<br>データ<br>トップマーク長<br>(サイクル) |
|---|---|---|---|---|---|---|---|---|---|---|
| X1 シリーズ | 2700 | 20H | 2EH | LONG<br>SHORT<br>LONG | 20H | 18H | 44H | 3CH | 03E8H | 0BB8H |
| MZ-2000<br>MZ-80B | 2000 | 80H | 41H | SHORT<br>LONG<br>SHORT | 2AH | 25H | 5AH | 55H | 2710H | 2AF8H |
| MZ-700/1200<br>MZ-80K/C | 1200 | 80H | 70H | SHORT<br>LONG<br>SHORT | 3FH | 3AH | 81H | 7CH | 55F5H | 2AF8H |

図6 7 ディレイ・カウンタ対応図

図6 8 トップマークの違い

リスト6-1

No. 1

```
100 '
110 '  TAPE FORMAT CHANGE
120 '              FOR X1 HuBASIC ( TAPE VERSION )
130 :
140 CLEAR &HFE3F:AD=&HFE40
150 FOR I=0 TO 9999
160   READ D$:IF D$="END" THEN 190
170   POKE AD+I,VAL("&H"+D$)
180 NEXT I
190 DATA E5,D5,C5,E5,21,60,FE,01
200 DATA 80,00,CD,41,00,D1,C1,F5
210 DATA ED,B0,F1,D1,E1,C9,END
220 :
230 CLS:PRINT,"TAPE FORMAT CHANGE":PRINT
240 PRINT "1)..MZ-700/80K/C  2)..MZ-2000/80B 3)..X1"
250 INPUT "LOAD FORMAT = ";LF$
260 ON VAL(LF$) GOSUB 310,340,410
270 INPUT "SAVE FORMAT = ";SF$
280 ON VAL(SF$) GOSUB 480,540,640
290 END
300 :
310 'MZ-700 LOAD
320 D=&H70:GOTO 360
330 :
340 'MZ-2000 LOAD
350 D=&H41
360 GOSUB 800
370 J1=&H40:J2=&HFE:GOSUB 750
380 F1=&H28:F2=&H20:GOSUB 830
390 RETURN
400 :
410 'X1 LOAD
420 D=&H2E:GOSUB 800
430 J1=&H41:J2=&H0 :GOSUB 750
440 F1=&H20:F2=&H28:GOSUB 830
450 RETURN
460 '-----------
470 :
480 'MZ-700 SAVE
490 S1=&H3F:S2=&H3A
500 L1=&H81:L2=&H7C
510 N3=&HF5:N4=&H55
520 GOTO 580
530 :
540 'MZ-2000 SAVE
550 S1=&H2A:S2=&H25
560 L1=&H5A:L2=&H55
570 N3=&H10:N4=&H27
580 N1=&HF8:N2=&H2A
590 GOSUB 860
600 J1=&H8A:J2=&HA5
610 GOSUB 930
620 N=&H80:GOTO 970
630 :
640 'X1 SAVE
650 S1=&H20:S2=&H18
660 L1=&H44:L2=&H3C
670 N1=&HB8:N2=&HB
680 N3=&HE8:N4=&H3
```

No. 2

```
690 GOSUB 860
700 J1=&HA5:J2=&H8A
710 GOSUB 930
720 N=&H20:GOTO 970
730 :
740 '-- Information Load Entry
750 POKE &H10C4,J1:POKE&H10C5,J2
760 POKE &H60B0,J1:POKE&H60B1,J2
770 POKE &H613C,J1:POKE&H613D,J2
780 RETURN
790 '-- Delay Counter  (Load)
800 POKE &HDC0,D
810 RETURN
820 '-- Top Make Logic (Load)
830 POKE &HD3B,F1:POKE &HD4E,F2
840 RETURN
850 '-- Delay Counter  (Save)
860 POKE &HD94,S1:POKE &HD9D,S2 'Short
870 POKE &HDAF,L1:POKE &HDB8,L2 'Long
880 '-- Top Make Lenght(Save)
890 POKE &HCE2,N1:POKE &HCE3,N2
900 POKE &HCEC,N3:POKE &HCED,N4
910 RETURN
920 '-- Top Make Logic (Save)
930 POKE &HCF0,J1:POKE &HD04,J1
940 POKE &HCFB,J2
950 RETURN
960 '-- Information Byte (Save)
970 POKE &H108B,N   'Monitor S Command
980 POKE &H608D,N   'SAVE Command
990 RETURN
```

# 6-5 バックアップ・ツール

IOCSを利用してマシン語およびBASICテキストのバックアップ・ツールをつくってみました（**リスト6-2**）。このプログラムではテープが入っていないとか，テープのつめが折れているかなどの細かいチェックは，すべてIOCSのルーチンに任せています。IOCSのサブルーチンについては，巻末付録を参照してください。

このプログラムは，1000Hから始まっていますが，入力するときにはE000Hから行います。誤りのないことを確認したらE003Hに実行を移してください。モニタのGコマンドでもHuBASIC のCALL命令のどちらでもかまいません。このプログラムには，自分自身をIPLから起動できるようにセーブするルーチンを含んでいます。できあがったテープはIPLからロードして実行してください。メッセージを表示するので使い方はわかると思います。

リスト6-2　　No. 1

```
  1:                  ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
  2:                  ;
  3:                  ;   CMT BACKUP TOOL
  4:                  ;
  5:                  ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
  6:                  ;
  7:                  ;
  8:                  ;  IOCS ENTRY
  9:                  ;
 10:                  ;
 11:    003B =        WRINF   EQU     003BH
 12:    003E =        WRDAT   EQU     003EH
 13:                  ;
 14:    0041 =        RDINF   EQU     0041H
 15:    0044 =        RDDAT   EQU     0044H
 16:                  ;
 17:    000B =        MSG     EQU     000BH
 18:    0013 =        PRNT    EQU     0013H
 19:    04C8 =        ACCDIS  EQU     04C8H
 20:    001B =        INKEY   EQU     001BH
 21:    0003 =        INPUTF  EQU     0003H
 22:    015A =        BINPUT  EQU     015AH
 23:    04A3 =        NL      EQU     04A3H
```

No. 2

```
24:   04A7 =         LETNL   EQU     04A7H
25:   04BA =         PRNTS   EQU     04BAH
26:   0DEA =         MSTOP   EQU     0DEAH
27:                  ;
28:   0026 =         COLORF  EQU     0026H
29:   0EA5 =         KBUFSW  EQU     0EA5H
30:   0EA6 =         POINT1  EQU     0EA6H
31:   0EA7 =         POINT2  EQU     0EA7H
32:                  ;
33:   1000                   ORG     01000H
34:                  ;
35:   1000 C32310    START:  JP      ST1
36:                  ;
37:   1003 =         IBUFF   EQU     $
38:   1003 =         MODE    EQU     $               ;file mode
39:   1003 C3                DEFB    0C3H            ;JP BLDIR
40:   1004 =         FNAME   EQU     $
41:   1004 A3E2              DEFW    BLDIR+0D000H
42:                  ;
43:   1006                   DEFS    16-2            ;file name
44:                  ;
45:   1014 00        PROT:   DEFB    00              ;protect mark
46:   1015 0000      LONG:   DEFW    0000            ;length
47:   1017 0000      ADR:    DEFW    0000            ;address
48:   1019 0000      EXEC:   DEFW    0000            ;exec adr.
49:   101B           DATE:   DEFS    8               ;date
50:                  ;
51:   1023 3100FF    ST1:    LD      SP,0FF00H       ;set stack pointer
52:   1026 210010            LD      HL,START        ;reset ENT.
53:   1029 222B01            LD      (012BH),HL
54:   102C 1100FE            LD      DE,0FE00H       ;memory max
55:   102F EB                EX      DE,HL
56:   1030 B7                OR      A
57:   1031 ED52              SBC     HL,DE
58:   1033 22A112            LD      (MAX),HL
59:                  ;
60:   1036 3E07              LD      A,7             ;white
61:   1038 322600            LD      (COLORF),A
62:                  ;
63:   103B 3E0C              LD      A,0CH           ;cls
64:   103D CD1300            CALL    PRNT
65:                  ;
66:   1040 CD5412    ST2:    CALL    MSGP
67:   1043 0D0D              DEFB    0DH,0DH
68:   1045 434D5420          DEFM    'CMT BACKUP TOOL Ver1.0A'
69:   105C 00                DEFB    0
70:                  ;
71:   105D CDA304    ST3:    CALL    NL
72:   1060 CD5412            CALL    MSGP
73:   1063 0D                DEFB    0DH
74:   1064 53455420          DEFM    'SET CASETTE TAPE IN DECK'
75:   107C 0D                DEFB    0DH
76:   107D 50555348          DEFM    'PUSH  [SPACE] KEY '
77:   108F 00                DEFB    0
78:                  ;
79:   1090 CD4012            CALL    SPINP
80:   1093 38AB              JR      C,ST2
81:                  ;
82:   1095 210310            LD      HL,IBUFF        ;inf. block adr.
```

```
 83:  1098 012000            LD    BC,0020H      ;length
 84:  109B CD4100            CALL  RDINF         ;read inf.
 85:  109E CD1612            CALL  ERR           ;err. check
 86:  10A1 38BA              JR    C,ST3
 87:                   ;
 88:  10A3 CDA311            CALL  IPRNT         ;print inf.
 89:                   ;
 90:  10A6 2AA112            LD    HL,(MAX)      ;check length
 91:  10A9 ED4B1510          LD    BC,(LONG)
 92:  10AD B7                OR    A             ;CF=0
 93:  10AE ED42              SBC   HL,BC
 94:  10B0 3015              JR    NC,ST4        ;OK.
 95:                   ;
 96:  10B2 CD5412            CALL  MSGP          ;memory err.
 97:  10B5 0D                DEFB  0DH
 98:  10B6 4D454D4F          DEFM  'MEMORY ERROR'
 99:  10C2 0D00              DEFB  0DH,0
100:  10C4 C35D10            JP    ST3
101:                   ;
102:  10C7 21A312      ST4:  LD    HL,PROGE      ;prog. load adr.
103:  10CA CD4400            CALL  RDDAT         ;read data block
104:  10CD CD1612            CALL  ERR           ;err. check
105:  10D0 DA5D10            JP    C,ST3         ;if err.
106:  10D3 CDEA0D            CALL  MSTOP
107:  10D6 CD5412            CALL  MSGP
108:  10D9 0D                DEFB  0DH
109:  10DA 52454144          DEFM  'READ OK.'
110:  10E2 0D00              DEFB  0DH,0
111:                   ;
112:  10E4 CD5412      ST5:  CALL  MSGP
113:  10E7 0D                DEFB  0DH
114:  10E8 50555348          DEFM  'PUSH [I] or [SP] or [E] KEY'
115:  1103 0D                DEFB  0DH
116:  1104 5B495D20          DEFM  '[I] =INFORMATION PRINT'
117:  111A 0D                DEFB  0DH
118:  111B 5B53505D          DEFM  '[SP]=WRITE TAPE'
119:  112A 0D                DEFB  0DH
120:  112B 5B455D20          DEFM  '[E] =NEW SOFT WARE SET '
121:  1142 00                DEFB  0
122:                   ;
123:  1143 CD4C12      ST6:  CALL  GETKY         ;key input
124:  1146 FE20              CP    20H           ;if [SP]
125:  1148 2812              JR    Z,ST7
126:  114A FE03              CP    3             ;if [BRK]
127:  114C 2896              JR    Z,ST5
128:  114E FE45              CP    'E'           ;if [END]
129:  1150 CA5D10            JP    Z,ST3
130:  1153 FE49              CP    'I'           ;print inf.
131:  1155 20EC              JR    NZ,ST6
132:                   ;
133:  1157 CDA311            CALL  IPRNT         ;print inf.
134:  115A 1888              JR    ST5
135:                   ;
136:  115C 210310      ST7:  LD    HL,IBUFF      ;inf. block adr.
137:  115F 012000            LD    BC,0020H      ;inf. length
138:  1162 CD3B00            CALL  WRINF         ;write inf.
139:  1165 CD1612            CALL  ERR           ;err. check
140:  1168 DAE410            JP    C,ST5         ;if err.
141:                   ;
```

No. 4

```
142:   116B 21A312            LD      HL,PROGE          ;prog. load adr.
143:   116E ED4B1510          LD      BC,(LONG)         ;prog. length
144:   1172 CD3E00            CALL    WRDAT             ;write data
145:   1175 CD1612            CALL    ERR               ;err. check
146:   1178 CDEA0D            CALL    MSTOP
147:   117B C3E410            JP      ST5               ;loop
148:                  ;
149:                  ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
150:                  ;       PRTHL
151:                  ;       print HL reg. width ASC
152:                  ;
153:                  ;       in:     HL=data
154:                  ;       out:    none
155:                  ;       dest.:  AF
156:                  ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
157:                  ;
158:   117E 7C        PRTHL:  LD      A,H               ;print high
159:   117F CD8311            CALL    PRTHX             ;print A
160:   1182 7D                LD      A,L               ;print low
161:                  ;
162:   1183 F5        PRTHX:  PUSH    AF                ;save AF
163:   1184 1F                RRA                       ;shift 4 bit
164:   1185 1F                RRA
165:   1186 1F                RRA
166:   1187 1F                RRA
167:   1188 CD8C11            CALL    ASC               ;print ASC
168:   118B F1                POP     AF
169:                  ;
170:   118C E60F      ASC:    AND     0FH               ;0000 1111b
171:   118E C630              ADD     A,30H
172:   1190 FE3A              CP      3AH
173:   1192 3802              JR      C,ASC1
174:   1194 C607              ADD     A,7               ;if A-F
175:   1196 CD1300    ASC1:   CALL    PRNT
176:   1199 C9                RET
177:                  ;
178:                  ;
179:   119A F5        BUFKIL: PUSH    AF
180:   119B 3AA60E            LD      A,(POINT1)
181:   119E 32A70E            LD      (POINT2),A
182:   11A1 F1                POP     AF
183:   11A2 C9                RET
184:                  ;
185:   11A3 CD5412    IPRNT:  CALL    MSGP              ;print inf.
186:   11A6 0D                DEFB    0DH
187:   11A7 4D4F4445          DEFM    'MODE '
188:   11AC 00                DEFB    0
189:                  ;
190:   11AD 110310            LD      DE,IBUFF          ;inf. block adr.
191:   11B0 1A                LD      A,(DE)            ;mode
192:   11B1 13                INC     DE
193:   11B2 CD8311            CALL    PRTHX
194:                  ;
195:   11B5 CD5412            CALL    MSGP
196:   11B8 0D                DEFB    0DH
197:   11B9 46494C45          DEFM    'FILE='
198:   11BE 00                DEFB    0
199:                  ;
200:   11BF 0610              LD      B,16              ;print name
```

```
201:    11C1 CD0E12             CALL    PRNTLP
202:                    ;
203:    11C4 CD5412             CALL    MSGP
204:    11C7 0D                 DEFB    0DH
205:    11C8 50415353           DEFM    'PASS='
206:    11CD 00                 DEFB    0
207:                    ;
208:    11CE 1A                 LD      A,(DE)
209:    11CF CD8311             CALL    PRTHX
210:                    ;
211:    11D2 CD5412             CALL    MSGP
212:    11D5 0D                 DEFB    0DH
213:    11D6 53544152           DEFM    'START ADR.='
214:    11E1 00                 DEFB    0
215:                    ;
216:    11E2 2A1710             LD      HL,(ADR)
217:    11E5 CD7E11             CALL    PRTHL
218:                    ;
219:    11E8 CD5412             CALL    MSGP
220:    11EB 0D                 DEFB    0DH
221:    11EC 4C454E47           DEFM    'LENGTH='
222:    11F3 00                 DEFB    0
223:                    ;
224:    11F4 2A1510             LD      HL,(LONG)
225:    11F7 CD7E11             CALL    PRTHL
226:                    ;
227:    11FA CD5412             CALL    MSGP
228:    11FD 0D                 DEFB    0DH
229:    11FE 45584543           DEFM    'EXEC='
230:    1203 00                 DEFB    0
231:                    ;
232:    1204 2A1910             LD      HL,(EXEC)
233:    1207 CD7E11             CALL    PRTHL
234:                    ;
235:    120A CDA704             CALL    LETNL
236:    120D C9                 RET
237:                    ;
238:    120E 1A         PRNTLP: LD      A,(DE)
239:    120F 13                 INC     DE
240:    1210 CDC804             CALL    ACCDIS
241:    1213 10F9               DJNZ    PRNTLP
242:    1215 C9                 RET
243:                    ;
244:    1216 B7         ERR:    OR      A
245:    1217 C8                 RET     Z
246:    1218 116312             LD      DE,ER1
247:    121B 3D                 DEC     A
248:    121C 2815               JR      Z,ERRE
249:    121E 116C12             LD      DE,ER2
250:    1221 3D                 DEC     A
251:    1222 280F               JR      Z,ERRE
252:    1224 117B12             LD      DE,ER3
253:    1227 3D                 DEC     A
254:    1228 2809               JR      Z,ERRE
255:    122A 118812             LD      DE,ER4
256:    122D 3D                 DEC     A
257:    122E 2803               JR      Z,ERRE
258:    1230 119712             LD      DE,ER5
259:    1233 CDA704     ERRE:   CALL    LETNL
```

No. 6

```
260:    1236 CD0B00             CALL    MSG
261:    1239 3E07               LD      A,7             :BELL
262:    123B CD1300             CALL    PRNT
263:    123E 37                 SCF                     ;CF=1
264:    123F C9                 RET
265:                    ;
266:    1240 CD4C12     SPINP:  CALL    GETKY           ;cur. inp.
267:    1243 FE20               CP      20H             ;if [SP]
268:    1245 C8                 RET     Z
269:    1246 FE03               CP      3               ;if [BRK]
270:    1248 20F6               JR      NZ,SPINP
271:    124A 37                 SCF                     ;CF=1
272:    124B C9                 RET
273:                    ;
274:    124C CD9A11     GETKY:  CALL    BUFKIL
275:    124F 3E01               LD      A,1             ;cur. inp.
276:    1251 C31B00             JP      INKEY
277:                    ;
278:    1254 E3         MSGP:   EX      (SP),HL         ;print from return adr.
279:    1255 F5                 PUSH    AF
280:    1256 7E         MSGP1:  LD      A,(HL)
281:    1257 23                 INC     HL
282:    1258 B7                 OR      A
283:    1259 2805               JR      Z,MSGP2         ;00 <-- end
284:    125B CD1300             CALL    PRNT
285:    125E 18F6               JR      MSGP1
286:    1260 F1         MSGP2:  POP     AF
287:    1261 E3                 EX      (SP),HL
288:    1262 C9                 RET
289:                    ;
290:    1263 42524541 ER1:      DEFM    'BREAK !!'
291:    126B 00                 DEFB    0
292:                    ;
293:    126C 43484543 ER2:      DEFM    'CHECK SUM ERR '
294:    127A 00                 DEFB    0
295:                    ;
296:    127B 53455420 ER3:      DEFM    'SET TAPE !! '
297:    1287 00                 DEFB    0
298:                    ;
299:    1288 57524954 ER4:      DEFM    'WRITE PROTECT '
300:    1296 00                 DEFB    0
301:                    ;
302:    1297 54415045 ER5:      DEFM    'TAPE END '
303:    12A0 00                 DEFB    0
304:                    ;
305:    12A1 0000       MAX:    DEFW    0000
306:                    ;
307:                    ;
308:    12A3 =          PROGE   EQU     $
309:                    ;
310:    12A3 3100E0     BLDIR:  LD      SP,0E000H
311:    12A6 2100E0             LD      HL,0E000H
312:    12A9 110010             LD      DE,START
313:    12AC 01A302             LD      BC,PROGE-START
314:    12AF EDB0               LDIR
315:    12B1 210010             LD      HL,START
316:    12B4 222B01             LD      (012BH),HL
317:                    ;
318:    12B7 3E0C               LD      A,0CH
```

```
319:   12B9 CD1300              CALL   PRNT
320:   12BC CDA304   BLDIR1:    CALL   NL
321:   12BF CD5412              CALL   MSGP
322:   12C2 53455420            DEFM   'SET TAPE'
323:   12CA 0D                  DEFB   0DH
324:   12CB 50555348            DEFM   'PUSH [SP] --> SAVE START '
325:   12E4 0D                  DEFB   0DH
326:   12E5 20202020            DEFM   '      [! ] --> BACKUP TOOL START'
327:   1304 0D                  DEFB   0DH
328:   1305 00                  DEFB   0
329:                 ;
330:   1306 CD4C12              CALL   GETKY
331:   1309 FE21                CP     '!'
332:   130B CA0000              JP     Z,0000
333:   130E FE20                CP     ' '
334:   1310 20AA                JR     NZ,BLDIR1
335:   1312 213913              LD     HL,PROGI
336:   1315 012000              LD     BC,0020H
337:   1318 CD3B00              CALL   WRINF
338:   131B CD1612              CALL   ERR
339:   131E 389C                JR     C,BLDIR1
340:   1320 210000              LD     HL,0000
341:   1323 01A312              LD     BC,PROGE-0000
342:   1326 CD3E00              CALL   WRDAT
343:   1329 CD1612              CALL   ERR
344:   132C 388E                JR     C,BLDIR1
345:   132E CD5412              CALL   MSGP
346:   1331 0D                  DEFB   0DH
347:   1332 4F4B2E              DEFM   'OK.'
348:   1335 0D00                DEFB   0DH,0
349:   1337 1883                JR     BLDIR1
350:                 ;
351:   1339 01       PROGI:     DEFB   01
352:   133A 4241434B            DEFM   'BACKUP TOOL  '
353:   1347 537973              DEFM   'Sys'
354:   134A 20                  DEFM   ' '
355:   134B A312                DEFW   PROGE-0000
356:   134D 0000                DEFW   0000
357:   134F 0000                DEFW   0000
358:   1351 00000000            DEFB   00,00,00,00,00
359:   1356 000000              DEFB   00,00,00
360:                 ;
361:   1359 =        BLDIRE     EQU    $
362:                 ;
363:   1359                     END
```

# 第7章
# フロッピーディスク



フロッピーディスクは，高速のロード，セーブができる他に，カセットテープではできないランダム・アクセス・ファイルを扱うことができ，ビジネス用途に威力を発揮します。

本章では，フロッピーディスク・コントローラなどあまりハード的なことにはふれずに，HuBASIC の管理を中心に解説します。

# 7-1 フロッピーディスク概要

X1のフロッピーディスクの仕様を**表7-1**に示します。フロッピーディスクのコントロールにはMB8877（富士通製）というLSIが使われています。このLSIはポピュラーなもので既に解説書も何冊か出ています。ここではMB8877（以下FDCと略す）についての詳しい解説はしないので他の文献を参考にしてください。**表7-2**はFDCに関するI/Oアドレス表です。なお，このうち0FFDH～0FFFHはX1 Turbo以外ではリザーブ領域となっておりユーザーが勝手に使うことは許されません。

**表7-1 ディスクの仕様**

| | |
|---|---|
| 記憶密度（Kバイト/ディスク）<br>アンフォーマット時<br>フォーマット時<br>（セクタ/トラック） | <br>500<br>327.6<br>（16） |
| 転送速度（Kビット/秒） | 250 |
| 記録密度（トラック/インチ） | 5876 |
| トラック密度（トラック/インチ） | 48 |
| トラック数 | 80 |
| 記録方式 | MFM |
| モータ起動時間（秒） | 1 |

表7-2 FDC の I/O アドレス

| アドレス | 入出力 | 内　　容 |
| --- | --- | --- |
| 0FF8 | OUT<br>IN | コマンド・レジスタ<br>ステータス・レジスタ |
| 0FF9 | I/O | トラック・レジスタ |
| 0FFA | I/O | セクタ・レジスタ |
| 0FFB | I/O | データ・レジスタ |
| 0FFC | OUT | ドライブ・セレクト，モータ ON/OFF |
|  | IN | 単密度記録（Turbo のみ） |
| 0FFD | OUT<br>IN | 倍密度記録（Turbo のみ） |
| 0FFE | OUT<br>IN | 高密度ディスク指定（Turbo のみ） |
| 0FFF | OUT<br>IN | FDC クロック周波数切り換え（500K/1M）<br>（Turbo のみ） |

# 7-2 HuBASICのディスク管理

HuBASICでは，1トラックを1クラスタとしユーザー作成ファイルの最小の管理単位としています。1トラックつまり1クラスタは，256バイトのセクタ16個から成っています。したがって，1クラスタは4Kバイトであり，たとえ1バイトのプログラムをセーブしても4Kバイトのエリアが使われることになります。

ディスクにはこういったユーザー用のエリアの他にディレクトリやFAT (File Allocation Table) といったシステムがファイルを管理するための特別な領域が設けられています。

ディレクトリにはファイル名やファイルが入っているクラスタ番号などが記録されています。ディレクトリの1個分のファイルの構造は32バイトの長さで**図7-1**のようになっています。これはカセットのファイル・インフォメーション・ブロックとほぼ同じような構造です。ディレクトをダンプした例を**図7-2**に示します。

**図7-1 ディレクトリの構造**

00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F 10 11 12 13 14 15 16 17 18 19 1A 1B 1C 1D 1E 1F

| モード | ファイル名 | 拡張子 | パスワード | データ長 L H | データ先頭アドレス L H | 実行アドレス L H | 年 月 曜日 日時分 | システム格納アドレス M L H |
|---|---|---|---|---|---|---|---|---|

○**モード ：ファイルの種類を表わす**

00 ：KILLされたファイル
FF ：使用ディレクトリ・テーブルの終わり
bit0＝1 ：Bin ファイル（機械語で書かれたファイル）
bit1＝1 ：Bas（BASIC テキストで書かれたファイル）
bit2＝1 ：Asc（ASCII セーブされたファイル）
bit4＝1 ：FILES で表示しない　0：表示する
bit5＝1 ：リードアフタライト ON　0：OFF
bit6＝1 ：書き込み禁止ファイル　0：書き込み OK
bit3, 7　予備

○**システム格納アドレス（3バイト）**

ファイル本体が格納されているクラスタ番号

**図7-2 ディレクトリのダンプ例**

```
#Device=1:              Record no.= 16
#Adr. =      HEX DATA                                              'Charactor code
#01000=11 42 41 53 49 43 20 43 5A 38 46 42 30 31 53 79 ' BASIC CZ8FB01Sy
#01010=73 20 00 A8 00 00 00 00 82 C1 27 17 58 00 02 00 's          ' X
#01020=02 53 74 61 72 74 20 75 70 20 20 20 20 20 42 61 ' Start up      Ba
#01030=73 20 05 01 00 00 00 00 82 C1 27 18 15 00 0D 00 's          '
#01040=42 55 74 69 6C 69 74 79 20 20 20 20 20 20 20 20 'BUtility
#01050=20 20 CD 12 00 00 00 00 82 C1 27 18 25 00 0E 00 '           ' %
#01060=41 55 74 69 6C 69 74 79 20 20 20 20 20 20 4F 62 'AUtility      Ob
#01070=6A 20 00 0A 00 F5 00 00 82 C1 27 18 34 00 10 00 'j          ' 4
#01080=02 50 53 47 20 54 52 41 49 4E 45 52 20 20 20 20 ' PSG TRAINER
#01090=20                   00 00 82 13 10 23 56 00 11 00 '            #V
                               20 20 20 20 20 20 20 ' CIRCLE1
                                  17 38 00 12 00 '  9          8
       02 4B 44 41 54 41 20               20 20 20 ' Graphic L&S
#01190=20 20 D0 00 00 00 00 00 85
#011A0=02 54 45 4D 50 20 20 20 20 20 20
#011B0=20 20 00 04 00 00 00 00 85 20 24 10 54                 $ T
#011C0=02 58 58 20 20 20 20 20 20 20 20 20 20 20 20 20 ' XX
#011D0=20 20 5B 00 00 00 00 00 85 21 25 17 49 00 1F 00 '  [        !% I
#011E0=00 59 20 20 20 20 20 20 20 20 20 20 20 20 20 20 ' Y
#011F0=20 20 86 00 00 00 00 00 85 35 01 15 33 00 20 00 '           5  3

#Device=1:              Record no.= 18       R...END
#Adr. =      HEX DATA                                              'Charactor code
#01200=FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF '
#01210=FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF '
#01220=FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF '
#01230=FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF '
#01240=FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF '
#01250=FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF '
#01260=FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF '
```

一方のFATはファイルごとのディスクの使用状態を示すものです。1クラスタは，4Kバイトの容量ですが，これを越えるファイルも存在し，この場合2つ以上のクラスタを使用して記録しなければなりません。ところが，ディレクトリ領域には，クラスタを指定する領域が1つしか用意されていません。そこで，FATはディスクの使用状態とともに，2つ以上のファイルをアクセルした場合の次のクラスタ番号を格納しています。FATのダンプ例を**図7-3**に示します。また，**表7-3**はFATデータの意味です。

**表7-4**はFAT，ディレクトリ，IPLの位置を示したものです。この表のレコード番号というのはディスクのトラックとセクタ番号を0からの通し番号にしたものです。たとえば，0トラック1セクタはレコード番号0，1トラック3セクタはレコード番号18にあたります。

**図7-3 FATのダンプ例**

```
#Device=1:              Record no.= 14       R...END
#Adr. =      HEX DATA                                        'Charactor code
#00E00=01 8F 03 04 05 06 07 08 09 0A 0B 0C 87 81 0F 82 '
#00E10=89 8B 81 83 82 83 83 82 19 1A 1B 1C 8E 80 83 80 '
#00E20=00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 '
#00E30=00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 '
#00E40=00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 '
#00E50=8F 8F 8F 8F 8F 8F 8F 8F 8F 8F 8F 8F 8F 8F 8F 8F '
#00E60=8F 8F 8F 8F 8F 8F 8F 8F 8F 8F 8F 8F 8F 8F 8F 8F '
#00E70=8F 8F 8F 8F 8F 8F 8F 8F 8F 8F 8F 8F 8F 8F 8F 8F '
#00E80=00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 '
#00E90=00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 '
#00EA0=00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 '
#00EB0=00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 '
#00EC0=00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 '
#00ED0=00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 '
#00EE0=00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 '
#00EF0=00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 '
```

**表7-3 FATデータの意味**

| | |
|---|---|
| 00 | 未使用領域 |
| 01 | システム |
| 02～7F | 次のクラスタ |
| 80～8F | クラスタの終わり（このデータから 7F を引いた値が最終クラスタの使用セクタ数） |
| 90～FF | 未使用 |

表7-4 **ディスクの使用状態**

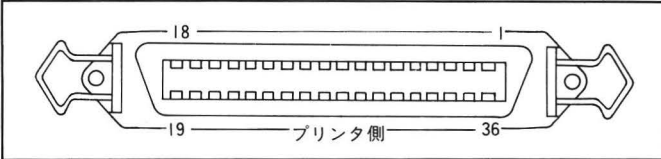
| クラスタ | トラック | セクタ | レコード | 用　途 |
|---|---|---|---|---|
| 0000 | 00 | 1 | 0 | IPL 用 |
| | | 2～14 | 1～ 13 | 未使用 |
| | | 15 | 14 | FAT |
| | | 16 | 15 | 未使用 |
| 0001 | 01 | 1～16 | 16～ 31 | ディレクトリ |
| 0002 | 02 | 1～16 | 32～47 | |
| 0003 | 03 | 1～16 | 48～63 | |
| 0004 | 04 | 1～16 | 64～79 | |
| ⋮ | ⋮ | ⋮ | ⋮ | |
| 000E | 0E | 1～16 | 224～ 239 | ユーザーエリア |
| 000F | 0F | 1～16 | 240～ 255 | |
| 0010 | 10 | 1～16 | 256～ 271 | |
| ⋮ | ⋮ | ⋮ | ⋮ | |
| 004D | 4D | 1～16 | 1232～1247 | |
| 004E | 4E | 1～16 | 1248～1263 | |
| 004F | 4F | 1～16 | 1264～1279 | |

# 第8章
# プリンタ



現在，市場に出回っているX1用プリンタは，シャープ純製品だけでも数種類あります。ここでは，これらを全てに対応することはできませんので，最初に発売されたCZ-800Pおよびこれと、コンパチビリティを持つEPSONのRP-80II(F/T)を想定して行います。

他のプリンタをお持ちの方でも，基本的なコントロール・コード体系やインターフェイス規格は同じなので参考になると思います。

# 8-1 セントロニクスについて

『セントロニクス』とは米国のセントロニクス社が考案したプリンタ・インターフェイス規格で，日本でももっとも普及しており，独占的な地位をしめています。X1でもセントロニクスに準拠した規格を採用しています。

この方式は基本的には8ビット・パラレル・データをハンド・シェイク方式で転送するものです。しかし，タイミングの微妙な違いから，同じセントロニクス準拠のプリンタといってもすべてX1につなげられるとは言いきれないようです。

**図8-1**がセントロニクス・インターフェイスの端子配列で，**図8-2**がX1のプリンタ端子配列です。

以下にセントロニクスの各信号の意味およびCZ-800Pでの違いを説明します。

**図8-1 セントロニクス端子**

| 端子番号 | 信号名 | 端子番号 | 信号名 |
|---|---|---|---|
| 1 | DATA STROBE | 19 | TWISTED PAIR GND |
| 2 | DATA 1 | 20 | 〃 |
| 3 | DATA 2 | 21 | 〃 |
| 4 | DATA 3 | 22 | 〃 |
| 5 | DATA 4 | 23 | 〃 |
| 6 | DATA 5 | 24 | 〃 |
| 7 | DATA 6 | 25 | 〃 |
| 8 | DATA 7 | 26 | 〃 |
| 9 | DATA 8 | 27 | 〃 |
| 10 | ACKNLG | 28 | 〃 |
| 11 | BUSY | 29 | 〃 |
| 12 | PE | 30 | 〃 |
| 13 | SLCT | 31 | INPUT PRIME |
| 14 | — | 32 | FAULT |
| 15 | OSCXT | 33 | LIGHT DETECTION |
| 16 | 0V | 34 | — |
| 17 | CHASSIS GND | 35 | — |
| 18 | ＋5V | 36 | — |

図8 2 **X1プリンタ端子**

| 13 | 11 | 9 | 7 | 5 | 3 | 1 |
|---|---|---|---|---|---|---|
| 14 | 12 | 10 | 8 | 6 | 4 | 2 |

X1側

| X1側 | | プリンタ側 |
|---|---|---|
| 端子番号 | 信号名 | 接続番号 |
| 1 | $\overline{\text{STROBE}}$ | 1 |
| 2 | PA 0 | 2 |
| 3 | PA 1 | 3 |
| 4 | PA 2 | 4 |
| 5 | PA 3 | 5 |
| 6 | PA 4 | 6 |
| 7 | PA 5 | 7 |
| 8 | PA 6 | 8 |
| 9 | PA 7 | 9 |
| 10 | アキ | — |
| 11 | BUSY | 11 |
| 12 | アキ | — |
| 13 | GND | 14 |
| 14 | GND | 16 |

**❶$\overline{\text{DATA STROBE}}$（センターマシン (C.M.) →プリンタ）**

プリンタがデータを読み込むためのストローブ・パルスです。定常状態ではHであり、Lに落ちた後にデータが読み込まれます。

最少でも0.5$\mu$sのパルス幅が必要です。

**❷〜❾ DATA 1〜8 (C.M.→プリンタ)**

パラレル・データで，Hであれば“1”, Lであれば“0”を示します。$\overline{\text{DATA STROBE}}$がLになる前から$\overline{\text{ACKNLG}}$がLになるまではホールドされなければなりません。CZ-800Pでの信号名はDATA BIT 1〜8。

**❿ $\overline{\text{ACKNLG}}$ (C.M.←プリンタ)**

プリンタが文字の入力を完了したときに出力するパルスで，約4〜10$\mu$sです。次のデータの転送要求パルスともいえます。CZ-800Pでの信号名は$\overline{\text{ACKNOWLEDGE}}$。

**⑪ BUSY（C.M.←プリンタ）**

プリンタが次のデータを取り込めるか否かの状態を示します。Lのときデータの取り込みが可能です。Hのときは次の場合が考えられます。

- ●データを取り込み中
- ●印字または改行復帰中
- ●ペーパーフィード中
- ●エラーが生じた場合

**⑫ PE（C.M.←プリンタ）**

用紙が無くなった場合にHになります。CZ-800Pでの信号名はPAPER END。

**⑬ SLCT（C.M.←プリンタ）**

プリンタの電源が入れられており，データ受信可能のときHを示します。CZ-800Pでの信号名はSELECT。

**⑭未使用**

CZ-800Pでは0 Vにつながれています。

**⑮ OSCXT**

CZ-800Pでは未使用。

**⑯ 0V**

ロジックのGNDレベルです。

**⑰ CHASSIS GND**

プリンタ・シャーシのGNDレベルです。

**⑱ ＋5V（C.M.←プリンタ）**

プリンタからの＋5V供給。CZ-800Pでは50mA MAX。

**⑲〜㉚ TWISTED PAIR GND**

向かい合った信号線とのツイストペア・リターン用GNDレベルで，ノイズ対策として用います。

**㉛ $\overline{\text{INPUT PRIME}}$（C.M.→プリンタ）**

定常状態はHで，Lにするとプリンタ・コントローラのイニシャライズを行います。プリント・バッファにデータがあった場合はクリアされます。

最小で50μs以上のパルス幅が必要です。

**㉜ $\overline{\text{FAULT}}$（C.M.←プリンタ）**

エラー発生時または用紙が無くなったときにLになります。

**㉝ LIGHT DETECTION**

CZ-800Pでは0Vにつながれています。

**㉞～㊱未使用**

# 8-2 プリンタとのハンドシェイク

ハンドシェイクとは相方が相手の動作終了を確認しながら行うデータ転送の方式で，非同期確認方式とも呼ばれています。

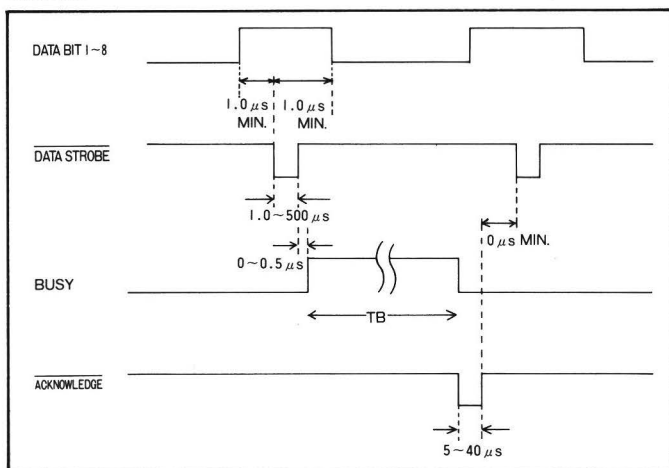
プリンタの場合，データ入力・改行・ホームフィードなどスピードの違う動作が混在しているため1ステップの動作時間が一定ではありません。ですから同期にたよらないハンドシェイク方式が適しているといえます。

セントロニクスの規格では，最低で1.5μsデータをバスに出力し，$\overline{\text{DATA STROBE}}$のパルス幅は最低で0.5μsです。

プリンタは$\overline{\text{DATA STROBE}}$がLになるとデータを取り込み，それを処理している間はBUSYにはHが出力されます。

処理が終了後，$\overline{\text{ACKNLG}}$に約4～10μs間Lが出力されて次のデータ要求を行います。

**図8-3**がCZ-800Pのタイミング図です。かなり余裕をもってセントロニクス規格を満たしています。

図8-3 **タイミング・チャート** (CZ-800P)

BUSY を H にしている時間 TB は最小で150μs，最大で(印字＋復帰＋改行) 時間と示されています。長くとも1回のフォーム・フィードの時間（約13秒) 以上BUSY が H のときはエラーが発生したとみなし，エラー処理へ飛ばすのが適切と思われます。

ちなみに，セントロニクスのハンドシェイク方式でデータを送るタイミングは，①BUSY が L に落ちた時点，②$\overline{\text{ACKNLG}}$ が L に落ちた時点の 2 通りの方法で知ることができますが，その両方を検知する必要はありません。

X 1 では BUSYによってタイミングを取っています。

## 8-2-1 ハンドシェイクの実際

X 1 ではプリンタ用に次のI/Oポートを使用しています。

| I/O アドレス | 内容 |
|---|---|
| 1 A00H　ビット 0 ～ビット 7 | 出力データ |
| 1 A01H　ビット 3 | BUSY |
| 1 A02H　ビット 7 | $\overline{\text{STROBE}}$ |

よって，実際のプログラムでは**リスト8-1**のように行います。これは PRNOUT という名前のサブルーチン形式にしてあります。

先に説明したように，約13秒以上 BUSY 信号が返ってこないときはエラーとみなし，キャリーを 1 にして復帰します。正常復帰は CY＝ 0 です。

さて，プリンタの電源が入っていないときにアクセスを行ってしまい，途中であわててプリンタの電源を入れた場合はどうなるでしょうか？　電源が OFF ならば BUSY 信号も L であり，X 1 は出力データを 1 バイト送ってしまいます。そのため，途中で電源を入れても最初の 1 文字は失なわれてしまい，印字されません。それに気づかずに長いプログラムを最後まで印字したが，最初の 1 文字が出なかったために打ち直し，ということが起きてしまいます。

このような失敗をふせぐためには，1行の先頭に00Hというコードを挿入しておきます。たったこれだけでもかなり使い勝手が良くなります。

リスト8-1 No. 1

```
 1:                         ;
 2:                         ;          PRINTER ACCESS
 3:                         ;                                 LIST 8-1
 4:    A000                            ORG     0A000H
 5:    A000 060B                       LD      B,11           ;String Length
 6:    A002 2111A0                     LD      HL,STRING-1    ;String Top Adrs
 7:    A005 C5          LOOP:          PUSH    BC
 8:    A006 7E                         LD      A,(HL)
 9:    A007 23                         INC     HL
10:    A008 CD1EA0                     CALL    PRNOUT         ;1Byte Print Out
11:    A00B C1                         POP     BC             ;       A = DATA
12:    A00C 380E                       JR      C,ERROR        ;CY = 1 THEN Error
13:    A00E 10F5                       DJNZ    LOOP
14:    A010 C9                         RET
15:
16:    A011 00                         DEFB    00
17:    A012 48656C6C STRING:           DEFM    'Hello  X1'
18:    A01B 0A                         DEFB    0AH
19:
20:    A01C 00          ERROR:         NOP                    ;Error Problem
21:    A01D C9                         RET
22:                         ;
23:    A01E F5          PRNOUT:        PUSH    AF
24:    A01F 010000                     LD      BC,0           ;Loop Counter
25:    A022 160B                       LD      D,11           ;Loop Counter
26:    A024 C5          BUSY:          PUSH    BC
27:    A025 01011A                     LD      BC,1A01H       ;8255 Port B
28:    A028 ED78                       IN      A,(C)
29:    A02A C1                         POP     BC
30:    A02B CB5F                       BIT     3,A            ;Bit 3 = Busy
31:    A02D 280B                       JR      Z,PRN
32:    A02F 05                         DEC     B
33:    A030 20F2                       JR      NZ,BUSY
34:    A032 0D                         DEC     C
35:    A033 20EF                       JR      NZ,BUSY
36:    A035 15                         DEC     D
37:    A036 20EC                       JR      NZ,BUSY
38:    A038 1815                       JR      ERR
39:
40:    A03A 01001A      PRN:           LD      BC,1A00H       ;8255 Port A
41:    A03D F1                         POP     AF
42:    A03E ED79                       OUT     (C),A          ;Data Out
43:
44:    A040 01021A                     LD      BC,1A02H       ;8255 Port C
45:    A043 ED78                       IN      A,(C)
46:    A045 CBBF                       RES     7,A            ;STROBE/ = 0
47:    A047 ED79                       OUT     (C),A
48:    A049 CBFF                       SET     7,A            ;STROBE/ = 1
49:    A04B ED79                       OUT     (C),A
50:    A04D B7                         OR      A              ;CY = 0
```

No. 2

```
51:    A04E C9                RET
52:
53:    A04F F1        ERR:    POP      AF
54:    A050 37                SCF                          ;CY = 1
55:    A051 C9                RET                          ;    'Device Off Line'
56:                   ;
57:    A052                   END
```

# 8-3 コントロール・コード体系

プリンタに送る印字データは通常はASCII コードです。しかし，印字データだけではプリンタの持っている各種の機能を使いきることができません。そこで，印字データの他に動作を行わせるコードが決められており，それをコントロール・コードと呼びます。

また，印字・改行といった基本動作より高度なことを行わせるには，ESC（1BH または27）につづく数バイトをコントロール・コードとして使用するのが一般的です。

基本的な動作（印字・改行等）はどのプリンタもほとんど統一されていますが，グラフィックの印字など高度な機能に関しては各社まちまちであるのが現状です。エプソンよりコントロール・コード体系の世界統一規格（ESC/P）が提案されているので，今後の動行に注目したいところです。

以下では，CZ-800P のコントロール・コードを説明します。

## 8-3-1 1バイト・コード

●**CR**（0DH）**キャリッジ・リターン**

CR のみ受信…………………何も動作しない。

データ＋CR 受信…………データを印字（改行はしない）。

●**LF**（0AH）**ライン・フィード**

LF のみ受信…………………1行改行。

データ＋LF 受信…………データを印字後，改行。

●**FF**（0CH）**ホーム・フィード**

プリント・バッファ内のデータを印字後，次のフォーマットの先頭まで用紙を送る。

●**CAN**（18H）**キャンセル**

プリント・バッファ内のデータをクリアする。ただし，ファンクション・コードは実行されるが，拡大文字状態は解除

される。

● **DC1** (11H)

プリンタをSELECT（受信可能状態）にする（初期状態)。

● **DC3** (13H)

プリンタをローカル(受信不可能状態)にする。以後はDC 1以外は受けつけない。

## 8-3-2 フィード,スキップ関係

● **ESC ＋6** (1 BH＋36H)

1/6インチの改行ピッチを指定する（初期状態)。

● **ESC ＋8** (1 BH ＋38H)

1/8インチの改行ピッチを指定する。

● **ESC ＋%＋9＋n** (1 BH ＋25H ＋39H ＋ n)

n/144インチの改行ピッチを指定する。nが0のときは無視される。

● **ESC ＋ VT ＋ $n_1$ ＋ $n_2$** (1 BH ＋ 0 BH ＋$n_1$ ＋ $n_2$)

$n_1$, $n_2$ により定められた行数だけ紙送りを行う。行数は$n_1$×10＋ $n_2$行となる。

● **ESC ＋ F ＋$n_1$ ＋ $n_2$** (1 BH ＋46H ＋ $n_1$ ＋ $n_2$)

ページ長を設定する。$n_1$, $n_2$は10進数で, ($n_1$ ×10＋ $n_2$)÷2インチに設定される。

● **ESC ＋5** (1 BH ＋ 35H)

現在のヘッドの位置がページの先頭になるようセットする。

● **DC4＋ {0またはn…} ＋?** (14H＋ {30Hまたはn…} ＋ 3FH)

垂直タブ位置を設定する。タブ位置でない行には0を置き，タブ位置に設定したい行にnを置く。nは1～14までで，チャンネルNo.を表す。

チャンネルNo.は下記の値で指定する。

| | | | | | |
|---|---|---|---|---|---|
| チャンネル1 | “1” | 31H | チャンネル8 | “8” | 38H |
| 〃 2 | “2” | 32H | 〃 9 | “9” | 39H |
| 〃 3 | “3” | 33H | 〃 10 | “:” | 3AH |
| 〃 4 | “4” | 34H | 〃 11 | “;” | 3BH |
| 〃 5 | “5” | 35H | 〃 12 | “<” | 3CH |
| 〃 6 | “6” | 36H | 〃 13 | “=” | 3DH |
| 〃 7 | “7” | 37H | 〃 14 | “>” | 3EH |

●**VT ＋ n**（0BH ＋ n）

垂直タブを実行する。n はチャンネル No.。

## 8-3-3 印字サイズ,印字数

●**ESC ＋ R**（1BH ＋ 52H）

10 CPI 文字（普通文字）を指定する（初期状態）。

●**ESC ＋ E**（1BH ＋ 45H）

12 CPI 文字（縮小文字）を指定する。

●**ESC ＋U**（1BH ＋ 55H）

横方向の 2 倍拡大文字を指定する。

10 CPI ベースの場合は 5 CPI 文字になり，12 CPI ベースの場合は 6 CPI 文字になる。行の印字途中でも文字単位で切換可能。

●**ESC ＋ A**（1BH ＋ 41H）

1 行の印字数をロング･ラインに切り換える（初期状態）。

1 行の印字数は次のように文字サイズによって異なる。

| 文字サイズ | 5 CPI | 6 CPI | 10CPI | 12CPI |
|---|---|---|---|---|
| 1 行の印字数 | 40字 | 48字 | 80字 | 96字 |

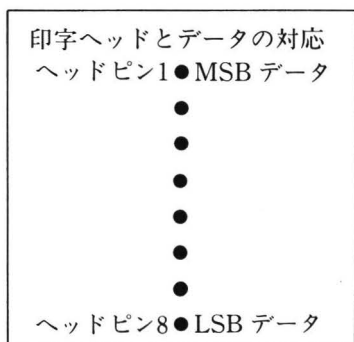
●**ESC ＋ B**（1BH ＋ 42H）

1 行の印字数をショート・ラインに切換える。

| 文字サイズ | 5 CPI | 6 CPI | 10CPI | 12CPI |
|---|---|---|---|---|
| 1 行の印字数 | 32字 | 38字 | 64字 | 76字 |

# 8-3-4 グラフィックの印字

● **ESC＋%＋2＋n1＋n2**（1BH＋25H＋32H＋$n_1$＋$n_2$）

グラフィック印字のデータ数を指定し，印字を開始する。$n_1$, $n_2$は16進で，データ数を示す。$n_1$の上位4ビットは常に0とみなされるので1行に印字できるデータの数は1024個。

指定されたデータ数だけグラフィック印字を行った後は通常の印字に戻る。1行中にキャラクタとグラフィックの混在が可能。

# 8-4 ハード・コピー

HuBASIC にはHCOPY という強力なハード・コピー命令が含まれていますが，これでハード・コピーを行うと画面の1ドットがプリンタでは4ドットに相当しているため大きな文字になってしまい，スピードも遅くなります。

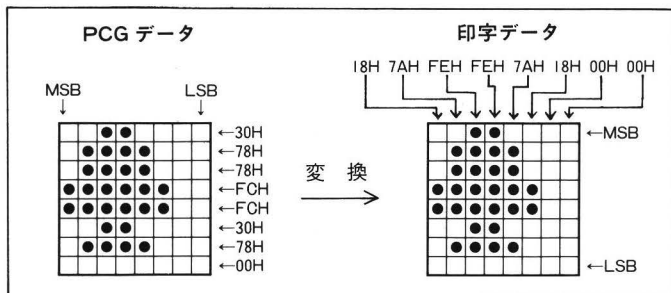
ここでは，2種類のハード・コピープログラムを紹介します。

## 8-4-1 キャラクタとPCG

**リスト8-2**がPCGを含めた文字のハード・コピープログラムです。プログラムを短くするためにIOCSを利用しています。

また，PCG RAMからPCGフォント・データを読み取って，ワーク・エリアでR・G・Bの各データのORをとっている点にも注目してください。さらにグラフィックも合成したい場合はキャラクタもフォント・データとして読み出しておくことが必要となります。このプログラムではハード・コピーのスピードを速くするためにキャラクタをグラフィック印字することは行っていません。

文字はそのままASCIIコードで送り，印字することができますが，PCGの文字はデータを1度変更してからグラフィック印字を行わなければなりません。

**図8-4 PCGデータ変換例**

リスト8-2

No. 1

```
 1:                      ;
 2:                      ;         HCOPY  Character & PCG
 3:                      ;                                  LIST 8-2
 4:                      ;
 5:                      ; --  Entry  --
 6:   0033 =             CGREAD  EQU     0033H              ;Read CG ROM/RAM
 7:   004A =             BRKCHK  EQU     004AH              ;Break Check
 8:   12D5 =             CR1LPL  EQU     12D5H
 9:   12DC =             ACCLPL  EQU     12DCH              ;Printer Out
10:                      ;
11:                      ; --  Work  --
12:   0007 =             WIDTH0  EQU     0007H
13:   00E9 =             INIADR  EQU     00E9H              ;VRAM Offset
14:                      ;
15:   C000                       ORG     0C000H
16:                      ;
17:   C000 3AE900                LD      A,(INIADR)         ;Offset Read
18:   C003 67                    LD      H,A
19:   C004 2E00                  LD      L,0
20:   C006 E5                    PUSH    HL
21:   C007 010030                LD      BC,3000H
22:   C00A 09                    ADD     HL,BC
23:   C00B EB                    EX      DE,HL              ;DE = Text Top Adrs
24:   C00C E1                    POP     HL
25:   C00D 0620                  LD      B,20H
26:   C00F 09                    ADD     HL,BC              ;HL = Attribute Top Adrs
27:   C010 CDD512                CALL    CR1LPL
28:   C013 0E19                  LD      C,25               ;Line Count Set
29:   C015 3A0700        LOOPY:  LD      A,(WIDTH0)         ;Column Count Set
30:   C018 47                    LD      B,A
31:   C019 C5            LOOPX:  PUSH    BC
32:   C01A 44                    LD      B,H
33:   C01B 4D                    LD      C,L
34:   C01C ED78                  IN      A,(C)
35:   C01E CB6F                  BIT     5,A                ;Bit 5 = 1 Then PCG RAM
36:   C020 201A                  JR      NZ,PCG
37:   C022 42                    LD      B,D
38:   C023 4B                    LD      C,E
39:   C024 ED78                  IN      A,(C)              ;A = ASCII
40:   C026 C1            PCGRET: POP     BC
41:   C027 13                    INC     DE
42:   C028 23                    INC     HL
43:   C029 CDDC12                CALL    ACCLPL             ;Printer Out
44:   C02C CD4A00                CALL    BRKCHK
45:   C02F CAD512                JP      Z,CR1LPL           ;Break Key In
46:   C032 10E5                  DJNZ    LOOPX
47:   C034 CDD512                CALL    CR1LPL
48:   C037 0D                    DEC     C
49:   C038 C215C0                JP      NZ,LOOPY
50:   C03B C9                    RET
51:                      ;
52:   C03C D5            PCG:    PUSH    DE                 ;PCG Print Out
53:   C03D E5                    PUSH    HL
54:   C03E 42                    LD      B,D
55:   C03F 4B                    LD      C,E
56:   C040 ED78                  IN      A,(C)              ;A = ASCII
57:   C042 21B6C0                LD      HL,WORK
58:   C045 0608                  LD      B,8
59:   C047 3600          LP:     LD      (HL),0             ;Work Clear
```

```
 60:   C049 23                INC    HL
 61:   C04A 10FB              DJNZ   LP
 62:   C04C 1E15              LD     E,15H          ;PCG Blue I/O Adrs
 63:   C04E CD8BC0            CALL   CGRD           ;Blue
 64:   C051 CD8BC0            CALL   CGRD           ;Red
 65:   C054 CD8BC0            CALL   CGRD           ;Green
 66:                  ;
 67:   C057 1608              LD     D,08H
 68:   C059 21AEC0            LD     HL,PCGDAT
 69:   C05C 01B6C0    LOOP1:  LD     BC,WORK        ;Convert 8Byte Data
 70:   C05F 1E08              LD     E,08H
 71:   C061 3600              LD     (HL),0
 72:   C063 0A        LOOP2:  LD     A,(BC)         ;1Byte Convert
 73:   C064 F5                PUSH   AF
 74:   C065 E680              AND    80H
 75:   C067 B6                OR     (HL)
 76:   C068 07                RLCA
 77:   C069 77                LD     (HL),A
 78:   C06A F1                POP    AF
 79:   C06B 07                RLCA
 80:   C06C 02                LD     (BC),A
 81:   C06D 03                INC    BC
 82:   C06E 1D                DEC    E
 83:   C06F C263C0            JP     NZ,LOOP2
 84:   C072 23                INC    HL
 85:   C073 15                DEC    D
 86:   C074 C25CC0            JP     NZ,LOOP1
 87:                  ;
 88:   C077 21A9C0            LD     HL,PCGDAT-5    ;Set Print Data Top
 89:   C07A 060C              LD     B,12           ;Data Length - 1
 90:   C07C 7E        LOOP3:  LD     A,(HL)
 91:   C07D CDDC12            CALL   ACCLPL         ;Printer Out
 92:   C080 23                INC    HL
 93:   C081 05                DEC    B
 94:   C082 C27CC0            JP     NZ,LOOP3
 95:   C085 7E                LD     A,(HL)         ;Last Data
 96:   C086 E1                POP    HL
 97:   C087 D1                POP    DE
 98:                  ;
 99:   C088 C326C0            JP     PCGRET         ;Return
100:                  ;
101:   C08B F5        CGRD:   PUSH   AF             ;PCG Data Read
102:   C08C D5                PUSH   DE
103:   C08D 21AEC0            LD     HL,PCGDAT
104:   C090 57                LD     D,A
105:   C091 CD3300            CALL   CGREAD         ;CG Data Read
106:   C094 D1                POP    DE
107:   C095 1C                INC    E
108:   C096 21B6C0    DATAOR: LD     HL,WORK        ;OR WORK , PCGDAT
109:   C099 01AEC0            LD     BC,PCGDAT
110:   C09C 1608              LD     D,8
111:   C09E 0A        LOOP4:  LD     A,(BC)
112:   C09F B6                OR     (HL)
113:   C0A0 77                LD     (HL),A
114:   C0A1 23                INC    HL
115:   C0A2 03                INC    BC
116:   C0A3 15                DEC    D
117:   C0A4 C29EC0            JP     NZ,LOOP4
118:   C0A7 F1                POP    AF
```

No. 3

```
119:   C0A8 C9                 RET
120:
121:                   ;
122:   C0A9 1B253200           DEFB     1BH,25H,32H,00H,08H
123:   C0AE            PCGDAT: DEFS     8
124:   C0B6            WORK:   DEFS     8
125:                   ;
126:   C0BE                    END
```

# 8-4-2 グラフィック

**リスト8-3**がGRAM 1 (Blue) のハード・コピーを行うプログラムです。GRAMの先頭よりラスタ順に8バイトづつワーク・エリアに取り出して，変換した後にグラフィック印字を行っています。

リスト8-3

No. 1

```
  1:                   ;
  2:                   ;        HCOPY  Graphic
  3:                   ;                                LIST 8-3
  4:                   ;
  5:                   ; -- Entry  --
  6:   004A =          BRKCHK  EQU      004AH           ;Break Check
  7:   12D5 =          CR1LPL  EQU      12D5H
  8:   12DC =          ACCLPL  EQU      12DCH           ;Printer Out
  9:                   ;
 10:                   ; -- Work  --
 11:   0007 =          WIDTH0  EQU      0007H
 12:   00E9 =          INIADR  EQU      00E9H           ;Offset
 13:                   ;
 14:   C000                    ORG      0C000H
 15:                   ;
 16:   C000 3AE900             LD       A,(INIADR)      ;Offset Read
 17:   C003 67                 LD       H,A
 18:   C004 2E00               LD       L,0
 19:   C006 010040             LD       BC,4000H
 20:   C009 09                 ADD      HL,BC
 21:   C00A 44                 LD       B,H             ;BC = Graphic Top Adrs
 22:   C00B 4D                 LD       C,L
 23:   C00C CDD512             CALL     CR1LPL
 24:   C00F 1619               LD       D,25            ;Line Count Set
 25:   C011 3A0700     LOOPY:  LD       A,(WIDTH0)      ;Column Count Set
 26:   C014 5F                 LD       E,A
 27:   C015 6F                 LD       L,A
 28:   C016 2600               LD       H,0
 29:   C018 29                 ADD      HL,HL
 30:   C019 29                 ADD      HL,HL
 31:   C01A 29                 ADD      HL,HL
 32:   C01B 7C                 LD       A,H
```

No. 2

```
33:   C01C 32A2C0          LD     (LENGTH),A     ;Bit Length Set
34:   C01F 7D              LD     A,L
35:   C020 32A3C0          LD     (LENGTH+1),A
36:   C023 C5              PUSH   BC
37:   C024 219BC0          LD     HL,CODE        ;Printer Code Out
38:   C027 0609            LD     B,9
39:   C029 7E      LOOP1:  LD     A,(HL)
40:   C02A CDDC12          CALL   ACCLPL
41:   C02D 23              INC    HL
42:   C02E 10F9            DJNZ   LOOP1
43:   C030 C1              POP    BC
44:                ;
45:   C031 C5      LOOPX:  PUSH   BC
46:   C032 D5              PUSH   DE
47:   C033 1E08            LD     E,8
48:   C035 2193C0          LD     HL,WORK
49:                ;
50:   C038 ED78    LOOP2:  IN     A,(C)
51:   C03A 77              LD     (HL),A         ;GRAM Data Set
52:   C03B 23              INC    HL
53:   C03C 78              LD     A,B
54:   C03D C608            ADD    A,08H
55:   C03F 47              LD     B,A
56:   C040 1D              DEC    E
57:   C041 C238C0          JP     NZ,LOOP2
58:                ;
59:   C044 0E08            LD     C,8
60:   C046 110080  LOOP3:  LD     DE,8000H       ;Convert 8Byte Data
61:   C049 2193C0          LD     HL,WORK
62:   C04C 0608            LD     B,8
63:   C04E 7E      LOOP4:  LD     A,(HL)         ;1Byte Convert
64:   C04F F5              PUSH   AF
65:   C050 A2              AND    D
66:   C051 B3              OR     E
67:   C052 07              RLCA
68:   C053 5F              LD     E,A
69:   C054 F1              POP    AF
70:   C055 07              RLCA
71:   C056 77              LD     (HL),A
72:   C057 23              INC    HL
73:   C058 10F4            DJNZ   LOOP4
74:   C05A 7B              LD     A,E
75:   C05B CDDC12          CALL   ACCLPL         ;Printer Out Bit Data
76:   C05E 0D              DEC    C
77:   C05F C246C0          JP     NZ,LOOP3
78:                ;
79:   C062 D1              POP    DE
80:   C063 C1              POP    BC
81:   C064 CD4A00          CALL   BRKCHK         ;Break Check
82:   C067 280D            JR     Z,BRK
83:   C069 03              INC    BC
84:   C06A 1D              DEC    E
85:   C06B C231C0          JP     NZ,LOOPX
86:                ;
87:   C06E CDD512          CALL   CR1LPL         ;On Other Line
88:   C071 15              DEC    D
89:   C072 C211C0          JP     NZ,LOOPY
90:   C075 C9              RET                   ;Return
91:                ;
```

No. 3

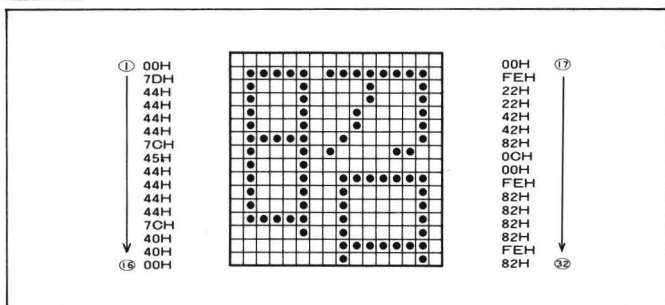
```
 92:  C076 AF        BRK:     XOR    A                  ;Break Plobrem
 93:  C077 D5        LOOP5:   PUSH   DE
 94:  C078 CDDC12             CALL   ACCLPL             ;Lest Bit
 95:  C07B D1                 POP    DE
 96:  C07C 1D                 DEC    E
 97:  C07D 20F8               JR     NZ,LOOP5
 98:  C07F 3A0700             LD     A,(WIDTH0)
 99:  C082 5F                 LD     E,A
100:  C083 15                 DEC    D
101:  C084 20F0               JR     NZ,BRK
102:  C086 CDD512             CALL   CR1LPL
103:  C089 3E1B               LD     A,1BH              ;Nonal Feed 1/6Inch
104:  C08B CDDC12             CALL   ACCLPL
105:  C08E 3E36               LD     A,36H
106:  C090 C3DC12             JP     ACCLPL             ;Break Return
107:                 ;
108:  C093           WORK:    DEFS   8
109:  C09B 1B25390E  CODE:    DEFB   1BH,25H,39H,0EH    ;Feed Inch = 14/144
110:  C09F 1B2532             DEFB   1BH,25H,32H        ;Printer Bit Image Code
111:  C0A2 0000      LENGTH:  DEFB   00,00              ;Bit length
112:                 ;
113:  C0A4                    END
```

320×200ドット表示モードでは縦横比が画面とプリンタとでほぼ同じになります（サークルを描くと良くわかります）。

GRAM 2，3と重ねてコピーを取りたい場合は，ワーク・エリア上で各データのORをとると良いでしょう。

# 8-5 漢字のプリント・アウト

漢字 ROM より読み出されたフォント・データは16×16ドット（32バイト）です。よって，プリンタではダブル・ストロークではじめて１行印字されます。

図8-5 漢字ROM フォント・データ格納順序

また，漢字データの格納順は**図8-5**に示すとおりで，これをプリンタ用のデータに変換しなければなりません。**リスト8-4**が漢字印字プログラムです。

リスト8-4 No. 1

```
 1:                     ;
 2:                     ;        KANJI Print Out
 3:                     ;                                 LIST 8-4
 4:                     ;
 5:                     ; --  Entry  --
 6:  004A =             BRKCHK  EQU     004AH             ;Break Check
 7:  12D5 =             CR1LPL  EQU     12D5H
 8:  12DC =             ACCLPL  EQU     12DCH             ;Printer Out
 9:                     ;
10:  C000                       ORG     0C000H
11:                     ;
12:  C000 216DC0                LD      HL,KDATA          ;HL = String Top
13:  C003 E5                    PUSH    HL
14:  C004 110800                LD      DE,0008H
15:  C007 19                    ADD     HL,DE
16:  C008 EB                    EX      DE,HL             ;DE = String Top + 8
17:  C009 E1                    POP     HL
18:  C00A 0E02                  LD      C,2
19:  C00C 3E02          LOOP1:  LD      A,2               ;A  = String Length
20:  C00E 87                    ADD     A,A
```

No. 2

```
21:   C00F 47                LD     B,A
22:   C010 C5                PUSH   BC
23:   C011 E5                PUSH   HL
24:   C012 2600              LD     H,0
25:   C014 68                LD     L,B
26:   C015 29                ADD    HL,HL
27:   C016 29                ADD    HL,HL
28:   C017 29                ADD    HL,HL
29:   C018 7C                LD     A,H
30:   C019 326BC0            LD     (LENGTH),A       ;Bit Length Set
31:   C01C 7D                LD     A,L
32:   C01D 326CC0            LD     (LENGTH+1),A
33:   C020 2164C0            LD     HL,CODE
34:   C023 0609              LD     B,9
35:   C025 7E       LOOP2:   LD     A,(HL)
36:   C026 CDDC12            CALL   ACCLPL
37:   C029 23                INC    HL
38:   C02A 10F9              DJNZ   LOOP2
39:   C02C E1                POP    HL
40:   C02D C1                POP    BC
41:   C02E C5       LOOP3:   PUSH   BC
42:   C02F D5                PUSH   DE
43:   C030 E5                PUSH   HL
44:   C031 0E08              LD     C,8
45:   C033 110080   LOOP4:   LD     DE,8000H         ;Convert 8Byte Data
46:   C036 0608              LD     B,8
47:   C038 E1                POP    HL
48:   C039 E5                PUSH   HL
49:   C03A 7E       LOOP5:   LD     A,(HL)           ;1Byte Convert
50:   C03B F5                PUSH   AF
51:   C03C A2                AND    D
52:   C03D B3                OR     E
53:   C03E 07                RLCA
54:   C03F 5F                LD     E,A
55:   C040 F1                POP    AF
56:   C041 07                RLCA
57:   C042 77                LD     (HL),A
58:   C043 23                INC    HL
59:   C044 10F4              DJNZ   LOOP5
60:   C046 7B                LD     A,E
61:   C047 CDDC12            CALL   ACCLPL           ;Printer Out Bit Data
62:   C04A 0D                DEC    C
63:   C04B C233C0            JP     NZ,LOOP4
64:   C04E E1                POP    HL
65:   C04F 111000            LD     DE,0010H
66:   C052 19                ADD    HL,DE            ;HL = HL + 16
67:   C053 D1                POP    DE
68:   C054 C1                POP    BC
69:   C055 10D7              DJNZ   LOOP3
70:   C057 CDD512            CALL   CR1LPL
71:   C05A CD4A00            CALL   BRKCHK           ;Break Check
72:   C05D C8                RET    Z
73:   C05E EB                EX     DE,HL
74:   C05F 0D                DEC    C
75:   C060 C20CC0            JP     NZ,LOOP1
76:   C063 C9                RET                     ;Return
77:                 ;
78:   C064 1B253910 CODE:    DEFB   1BH,25H,39H,10H  ;Feed Inch = 16/144
79:   C068 1B2532            DEFB   1BH,25H,32H      ;Printer Bit Image Code
```

```
 80:    C06B 0000      LENGTH: DEFB    00,00            ;Bit length
 81:                   ;
 82:    C06D 007D4444  KDATA:  DEFB    00H,7DH,44H,44H  ;KANJI 'SHYO' Data
 83:    C071 44447C45          DEFB    44H,44H,7CH,45H
 84:    C075 44444444          DEFB    44H,44H,44H,44H
 85:    C079 7C404000          DEFB    7CH,40H,40H,00H
 86:    C07D 00FE2222          DEFB    00H,0FEH,22H,22H
 87:    C081 4242820C          DEFB    42H,42H,82H,0CH
 88:    C085 00FE8282          DEFB    00H,0FEH,82H,82H
 89:    C089 8282FE82          DEFB    82H,82H,0FEH,82H
 90:                   ;
 91:    C08D 037C0808          DEFB    03H,7CH,08H,08H  ;KANJI 'WA' Data
 92:    C091 087F0818          DEFB    08H,7FH,08H,18H
 93:    C095 1C2A2A48          DEFB    1CH,2AH,2AH,48H
 94:    C099 08080808          DEFB    08H,08H,08H,08H
 95:    C09D 0000007E          DEFB    00H,00H,00H.7EH
 96:    C0A1 42424242          DEFB    42H,42H,42H,42H
 97:    C0A5 42424242          DEFB    42H,42H,42H,42H
 98:    C0A9 427E4200          DEFB    42H,7EH,42H,00H
 99:                   ;
100:    C0AD                   END
```

漢字 ROM からの読み出しは省略しています。すでにワーク・エリアに漢字データが1行分セットされているものとします。ここでは例として『昭和』の2文字分のデータを使用しています。

このプログラムではワーク・エリアにセットされた漢字ROMと同じフォーマットのデータを破壊せずに1行分印字しています。

ただし，**リスト8-4**での印字は**図8-6**の通り，かなり大きな字体になってしまいます。

プリンタのフィード量調節と縮小文字機能を使って，通常の印字と同じぐらいの大きさで漢字を打たせることができます。そのためのプログラムが，**リスト8-5**です。

**図8-6 リスト8-4の漢字印字例(原寸)**

昭和

リスト8-5

No. 1

```
 1:                      ;
 2:                      ;          KANJI Print Out 2
 3:                      ;                                    LIST 8-5
 4:                      ;
 5:                      ; --  Entry  --
 6:   004A =             BRKCHK  EQU      004AH             ;Break Check
 7:   12D5 =             CR1LPL  EQU      12D5H
 8:   12DC =             ACCLPL  EQU      12DCH             ;Printer Out
 9:                      ;
10:   C000                       ORG      0C000H
11:                      ;
12:   C000 219BC0                LD       HL,KDATA          ;HL = String Top
13:   C003 3E02                  LD       A,2               ;A  = String Length
14:   C005 CD0EC0                CALL     KANJI
15:   C008 2803                  JR       Z.BRK             ;ZF =1 Then Break
16:   C00A CD0EC0                CALL     KANJI
17:   C00D C9            BRK:    RET
18:                      ;
19:   C00E F5            KANJI:  PUSH     AF
20:   C00F E5                    PUSH     HL
21:   C010 87                    ADD      A,A
22:   C011 CA84C0                JP       Z,EXT
23:                      ;
24:   C014 E5                    PUSH     HL
25:   C015 3243C0                LD       (COUNT+1),A
26:   C018 2600                  LD       H,0
27:   C01A 6F                    LD       L,A
28:   C01B 29                    ADD      HL,HL
29:   C01C 29                    ADD      HL,HL
30:   C01D 29                    ADD      HL,HL
31:   C01E 7C                    LD       A,H
32:   C01F 3299C0                LD       (LENGTH),A        ;Bit Length Set
33:   C022 7D                    LD       A,L
34:   C023 329AC0                LD       (LENGTH+1),A
35:   C026 E1                    POP      HL
36:                      ;
37:   C027 1190C0                LD       DE,ESCF           ;2/144 Feed & Elite Size
38:                      ;
39:   C02A 0606                  LD       B,6
40:   C02C 1A            LOOP1:  LD       A,(DE)
41:   C02D CDDC12                CALL     ACCLPL
42:   C030 13                    INC      DE
43:   C031 10F9                  DJNZ     LOOP1
44:                      ;                                  ;--
45:   C033 0602                  LD       B,2
46:   C035 C5            LOOP2:  PUSH     BC
47:                      ;
48:   C036 1196C0                LD       DE,ESCG           ;Bit Image Mode
49:   C039 0605                  LD       B,5
50:   C03B 1A            LOOP:   LD       A,(DE)
51:   C03C CDDC12                CALL     ACCLPL
52:   C03F 13                    INC      DE
53:   C040 10F9                  DJNZ     LOOP
54:                      ;
55:   C042 0600          COUNT:  LD       B,0
56:   C044 C5            LOOP3:  PUSH     BC                ;----
57:   C045 E5                    PUSH     HL
58:   C046 0E08                  LD       C,8
59:   C048 110080        LOOP4:  LD       DE,8000H          ;Convert 8byte Data
```

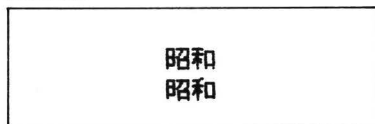
```
 60:   C04B 0608                LD     B,8
 61:   C04D E1                  POP    HL
 62:   C04E E5                  PUSH   HL
 63:   C04F 7E        LOOP5:    LD     A,(HL)          ;1Byte Convert
 64:   C050 F5                  PUSH   AF
 65:   C051 A2                  AND    D
 66:   C052 B3                  OR     E
 67:   C053 07                  RLCA
 68:   C054 5F                  LD     E,A
 69:   C055 F1                  POP    AF
 70:   C056 07                  RLCA
 71:   C057 77                  LD     (HL),A
 72:   C058 23                  INC    HL
 73:   C059 23                  INC    HL
 74:   C05A 10F3                DJNZ   LOOP5
 75:   C05C 7B                  LD     A,E
 76:   C05D CDDC12              CALL   ACCLPL          ;Printer Out Bit Data
 77:   C060 0D                  DEC    C
 78:   C061 C248C0              JP     NZ,LOOP4
 79:   C064 E1                  POP    HL
 80:   C065 111000              LD     DE,0010H
 81:   C068 19                  ADD    HL,DE           ;HL = HL +16
 82:   C069 C1                  POP    BC
 83:   C06A 10D8                DJNZ   LOOP3
 84:                  ;                                ;----
 85:   C06C CDD512              CALL   CR1LPL
 86:   C06F C1                  POP    BC
 87:   C070 E1                  POP    HL
 88:   C071 E5                  PUSH   HL
 89:   C072 23                  INC    HL
 90:   C073 10C0                DJNZ   LOOP2
 91:                  ;                                ;--
 92:   C075 118CC0              LD     DE,ESCN         ;Normal Feed & Size
 93:   C078 0604                LD     B,4
 94:   C07A 1A        LOOP6:    LD     A,(DE)
 95:   C07B CDDC12              CALL   ACCLPL
 96:   C07E 13                  INC    DE
 97:   C07F 10F9                DJNZ   LOOP6
 98:   C081 CDD512              CALL   CR1LPL
 99:                  ;
100:   C084 E1        EXT:      POP    HL
101:   C085 F1                  POP    AF
102:   C086 57                  LD     D,A
103:   C087 CD4A00              CALL   BRKCHK
104:   C08A 7A                  LD     A,D
105:   C08B C9                  RET
106:                  ;
107:   C08C 1B361B52  ESCN:     DEFB   1BH,36H,1BH,52H ;Normal
108:   C090 1B253902  ESCF:     DEFB   1BH,25H,39H,02H ;Feed Inch = 2/144
109:   C094 1B45                DEFB   1BH,45H         ;Elite Size
110:   C096 1B2532    ESCG:     DEFB   1BH,25H,32H     ;Printer Bit Image Code
111:   C099 0000      LENGTH:   DEFB   0,0             ;Bit Length
112:                  ;
113:   C09B 007D4444  KDATA:    DEFB   00H,7DH,44H,44H ;KANJI 'SHYO' Data
114:   C09F 44447C45            DEFB   44H,44H,7CH,45H
115:   C0A3 44444444            DEFB   44H,44H,44H,44H
116:   C0A7 7C404000            DEFB   7CH,40H,40H,00H
117:   C0AB 00FE2222            DEFB   00H,0FEH,22H,22H
118:   C0AF 4242820C            DEFB   42H,42H,82H,0CH
```

No. 3

```
119:    C0B3 00FE8282           DEFB    00H,0FEH,82H,82H
120:    C0B7 8282FE82           DEFB    82H,82H,0FEH,82H
121:                 ;
122:    C0BB 037C0808           DEFB    03H,7CH,08H,08H ;KANJI  'WA' Data
123:    C0BF 087F0818           DEFB    08H,7FH,08H,18H
124:    C0C3 1C2A2A48           DEFB    1CH,2AH,2AH,48H
125:    C0C7 08080808           DEFB    08H,08H,08H,08H
126:    C0CB 0000007E           DEFB    00H,00H,00H,7EH
127:    C0CF 42424242           DEFB    42H,42H,42H,42H
128:    C0D3 42424242           DEFB    42H,42H,42H,42H
129:    C0D7 427E4200           DEFB    42H,7EH,42H,00H
130:                 ;
131:    C0DB                    END
```

こちらは完全なサブルーチン形式になっており，HL レジスタに漢字データの先頭アドレスを，A レジスタに1行の字数を入れてコールします。この例では『昭和』という漢字を2行出力してみました（**図8-7**）。

**図8-7 リスト8-5の漢字印字例（原寸）**

昭和
昭和

9 ピンのプリンタでもソフトウェアの工夫しだいで，これだけの印字品質で漢字印字が可能となります。

# 第9章
# IPL ROMの解析・活用



**IPL (Initial Program Loader) は，通常電源投入時に1度だけ実行されて，外部ファイル（テープやディスクなど）からシステム・プログラム（BASICがその代表）を読み込み，制御をそれに移します。X1のIPLにはこの他にテレビタイマー用のプログラムも入っていますが，中には汎用的に使えるサブルーチンもあるので，ここではその解析と活用例について述べていきます。**

# 9-1 IPL ROM 概要

IPL は電源投入時に実行され，外部ファイルからシステム・プログラムを読み込みます。ですから，システム・プログラムを自作しようと思った場合，IPL ROM の管理も重要になってきます。

また，X1 の IPL ROM にはローダープログラムの他に簡単なテレビタイマーも含まれており，X1 得意のテレビ制御関係のプログラムをつくるときには十分に利用価値のあるものと思われます。その他，テープやディスクのアクセス，1 文字キー入力，画面 1 文字出力，16進表示など有益なサブルーチンも含まれています。

## 9-1-1 IPL ROM の起動

IPL ROM はアドレスの 0000H～0FFFH に割り当てられていて，電源を入れた時点にアクセスされ，IPL プログラムが実行されます。そして，外部記憶装置からシステム・プログラムの読み込みが終了した時点で切り離され，システム・プログラムに実行が移されます。よって，BASIC などのシステムが読み込まれた後，IPL ROM はアドレスのどこにも存在しません。

ところで，システム・プログラムのほとんどはアドレスの 0000H から始っています。IPL ROM も同じアドレスを有していたのでは読み込んだプログラムをメモリに格納することができないはずです。

X1 ではこれを次のような方法で同一のアドレスへのシス

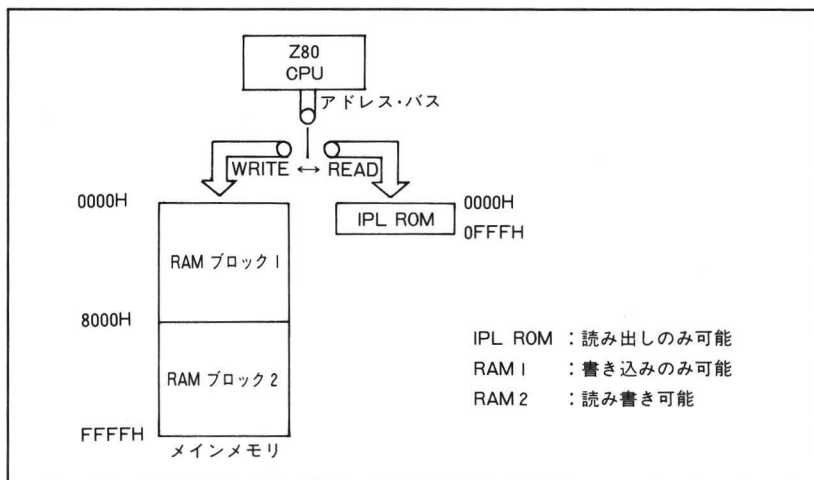
図9-1 **IPLのアクセス**

テム・プログラムの格納を可能にしています。

**図9-1**に示すように，メモリのREAD時にはIPL ROMがアクセスされ，WRITE時にはRAMがアクセスされるわけです。IPL ROMは読み出すことさえできれば，そのプログラムを実行できますし，ROMですから当然書き込む必要などないのです。うまい方法を考えたものです。

IPLが起動すると8255, CRTCなどの初期設定，スタック・ポインタ，文字の色などの初期設定を行い，次にキーが押されているかどうかを調べます。

このとき押されているキーによって外部デバイスが決まります（**図9-2**）。意味のないキーの場合は無視されます。

図9-2 **外部デバイスの選択**

| 入力キー | 動　作 |
|---|---|
| F | フロッピーディスクからシステムをロード |
| R | 外部 ROM　　　〃 |
| C | カセットテープ　　〃 |
| T | タイマーセットへジャンプする。 |
| SHIFT + BREAK | IPL メニューへ戻る。 |

キーが押されていなかった場合は，各外部デバイスから自動的に読み込みを開始します。このときの優先順位は，①フロッピーディスク，②外部ROM，③カセットテープで，この順に接続されているかどうかをチェックしていき，つながっていた場合のみインフォメーション・ブロックを読み込みます。そして，マシン語のファイルのとき，いよいよプログラム本体を読み込みます。

インフォメーション・ブロックのフォーマットはどのデバイスでも共通であり，X1であつかうファイル・インフォメーションはBASICファイルも含めて，すべて統一されています。**図9-3**にその詳しい構成を示します。

図9 3 **ファイル・インフォメーション・ブロック**

| アドレス | 内容 | | 説明 |
|---|---|---|---|
| FF00 | モード | | ●ファイルの種類を表わす<br>1.フロッピーディスクの場合<br>00は KILL されたファイル<br>FF は使用ディレクトリ・テーブルの終わり。<br>ビット0が1… Bin ファイル(マシン語で書かれたファイル)<br>ビット1が1… Bas ファイル(BASIC テキストで書かれたファイル)<br>ビット2が1… Asc ファイル(ASCII セーブされたファイル)<br>ビット4が1… FILES で表示しない：0…表示する。<br>ビット5が1…リード・アフターライト ON：0… OFF<br>ビット6が1…書き込み禁止ファイル：0…書き込み OK<br>ビット3とビット7は予備<br>(注)カセット，ROM の　場合ビット0～2までフロッピーディスクと同じだが，ビット3～7は未使用。 |
| FF01～FF0D | ファイル名 | | ●ファイル名(13文字) |
| FF0E～FF10 | 拡張子 | | ●ユーザー指定拡張子エリア(3文字) |
| FF11 | パスワード | | ● パスワード無指定の場合20H を入れる。 |
| FF12 | データ長 | (L) | ●プログラムの長さをバイト数で示す。<br>ただし，マシン語ファイルのみ有効となる。 |
| FF13 | | (H) | |
| FF14 | データ先頭アドレス | (L) | ●ロード時のメイン・メモリ先頭アドレスが入る。ただし，マシン語ファイルのみ有効。 |
| FF15 | | (H) | |
| FF16 | 実行アドレス | (L) | ●ロードされたプログラムの実行開始番地をメイン・メモリ上のアドレスで指定。 |
| FF17 | | (H) | |
| FF18 | 作成年月日 | (年) | ●ファイルが作成された年，月，曜日，日付，時刻（時，分）が入る。 |
| FF19 | | (月・曜日) | |
| FF1A | | (日) | |
| FF1B | | (時) | |
| FF1C | | (分) | |
| FF1D～FF1F | システム格納アドレス | | ●外部デバイス上のどこのアドレスから，ファイル本体が格納されているかを示す。カセットテープの場合常に00が格納される。 |

## 9-1-2 IPL ROM へのアクセス

システムが起動した直後，IPL ROM は切り離されてしまいますので，IPL 内のプログラムを活用しようとした場合，再度 IPL ROM をつながなければなりません。

そのためのスイッチは，I/O アドレスの 1D ＊＊ H に割り当てられています（下位アドレスは何でも良い）。

このI/O アドレスに何かを書き込む動作を行うと，IPL ROM がメイン・メモリ上の0000H～0FFFHにつながれます。

逆に IPL ROM を切り離すには I/O アドレスの 1E ＊＊ H に値を書き込む動作を行います。このプログラムは次のようになります。

● **IPL ROM 接続**

```
LD     B, 1DH …………下位アドレスは何でもよい
OUT    (C), A…A レジスタの値も何でもかまわない
```

● **IPL ROM 切り離し**

```
LD     B, 1EH
OUT    (C), A
```

また，B レジスタを壊したくない場合は次の方法が使えます。

● **IPL ROM 接続**

```
LD     A, 1DH
OUT    (00H), A…00Hの値は何でもよい
```

● **IPL ROM 切り離し**

```
LD     A, 1EH
OUT    (00H), A
```

IPL が接続された後に，任意のエントリーアドレスへジャンプまたはコールすることで IPL のプログラムを活用できます。

# 9-2 IPL ROMの解析

IOCSの解析は各書籍で取り上げられていますが，IPLについての解析はあまり見かけません。ここでは有益と思われるサブルーチンとその内容を説明します。これを機会にX1唯一（ターボをのぞく）のプログラムROMであるIPLにも目を向けてみてください。

**表9-1 IPL ROMエントリ表**

| アドレス | 内　容 |
|---|---|
| 0000H | IPL コールド・スタート　(JP　0006H) |
| 0003H | TV タイマースタート　(JP　0700H) |
| 0006H | IPL コールド・スタート1<br>●時間待ちループ |
| 0015H | IPL コールド・スタート2<br>●スタック・ポインタ初期設定 (LD　SP, 0)<br>●8255②モード・セット　出力値82H<br>モード0，ポートA＝出力，ポートB＝入力，ポートC＝出力<br>●8255②ポートC 初期設定　出力値72H<br>40字モード，スムーズ・スクロール OFF<br>●CRTC 初期設定　(CALL　0451H) |
| 0020H | IPL リスタート<br>●画面クリア<br>●キーセンス<br>●ディスク・センス & ロード　(CALL　010EH)<br>●ROM センス & ロード　(CALL　01A1H) |
| 0052H～0082H | テープ・センス & ロード<br>●テープのインフォメーション部を読んで，もしマシン語ファイルであればロードし実行する。その他のファイルならエラールーチンへジャンプ。 |
| 0066H | リセット・ボタンが押されたとき，このアドレスより実行される。 |
| 0083H | IPL ROM からの飛び出し<br>●IPL ROM を切り離し，FF16，17H に格納されているアドレスにジャンプする。 |

| アドレス | 内　容 |
|---|---|
| 0094H～00BFH | ROM センス & ロード |
| 00C5H～00EAH | テープからのロードが決定したときの処理<br>●テープ・センス。もし，テープがセットされていないときはメッセージを表示して待機。 |
| 00EBH | ファイルモード・エラー表示 1（カセット停止） |
| 00F0H | ファイルモード・エラー表示 2 |
| 00F5H | NOT READY エラー表示 |
| 00FAH | LOAD エラー表示 |
| 00FDH | エラーメッセージ表示<br>●DE レジスタに文字列先頭アドレスを入れて，ここへジャンプする。表示後キーセンスへ移る。 |
| 010EH～013FH | キーセンス<br>●[F]または[ハ]キー入力　(JP　017AH)<br>●[R]　〃　[ス]　〃　(JP　0094H)<br>●[C]　〃　[ソ]　〃　(JP　0052H)<br>●[T]　〃　[カ]　〃　(JP　00C0H)<br>●[SHIFT]+[BREAK]　(JP　00F5H) |
| 0140H | IPL ROM 切り離しと HL レジスタのアドレスへジャンプ |
| 017AH | キー指定でのディスク・ロード<br>●ドライブ番号入力処理 |
| 01A1H | ディスク・センス & ロード<br>●ディレクトリのロードとチェック<br>●マシン語ファイルであればロードし実行 |
| 033BH～0346H | ディスク・エラー処理 |

**表9-2 IPL ROM サブルーチン表**

| アドレス | 内　容 |
|---|---|
| 0146H | ROM からのロード・サブルーチル<br>●ROM ボードからメイン・メモリにプログラムをロードする。HL にロード先アドレス，BC に転送バイト数を入れてコールする。 |
| 0153H | ROM センス・サブルーチン<br>●ROM ボードがつながっていなければ，ゼロフラグがリセットされて返る。 |
| 016FH | “ IPL is looking for a ” 表示サブルーチン<br>●DE にファイル・ネーム格納先頭アドレス |

| アドレス | 内　容 |
| --- | --- |
| 01EDH | ディレクトリ・ロード・サブルーチン<br>●出力：キャリーフラグが0でOK，1のときエラー |
| 0202H | ディレクトリ・チェック・サブルーチン<br>●ファイルが機械語でかつSYSファイルであるかのチェック<br>入力：HLにインフォメーション先頭アドレス<br>出力：ゼロフラグ・セットのときOK，リセットのときエラー |
| 021AH | ディスクからのデータ・ロード・サブルーチン<br>●入力　HLにロード先アドレス<br>DEにレコード番号<br>Aにセクター番号 |
| 02BCH | ディスク・モータ停止サブルーチン |
| 02D1H | ディスク・モータ起動サブルーチン |
| 0318H | ディスク・シーク・サブルーチン |
| 0347H | 時間待ちサブルーチン<br>●27．25μ秒＋(Aレジスター1)×4.5μ秒の遅延。ただし，Aレジスタが0のときは256とみなす。<br>レジスタ：全レジスタ保存<br>入力：Aレジスタにループ・カウンタ |
| 0355H | 1文字キー入力（カーソル点滅まち）サブルーチン<br>●レジスタ：Aレジスタ以外保存<br>出力：Aレジスタに入力キーのASCIIコード |
| 038AH | 1文字キー入力（リアルタイム）サブルーチン<br>●レジスタ：Aレジスタ以外保存<br>出力：AレジスタおよびFF85Hに入力キーのASCIIコード |
| 03CBH | 文字列表示サブルーチン<br>●DEレジスタが示すアドレスからのデータを画面に表示する。データエンドは00H。<br>レジスタ：A，DEレジスタ以外は保存<br>入力：DEレジスタに文字列先頭アドレス |
| 03D4H | “XX”と表示<br>●レジスタ：Aレジスタ以外保存 |
| 03D9H | 1文字画面出力サブルーチン<br>●Aレジスタの値をASCIIコードとみなして画面に出力。ただし19H以下はコントロール・コードとみなしてその処理を実行する。<br>レジスタ：全てのレジスタを保存<br>入力：Aレジスタに出力するASCIIコード |

| アドレス | 内　容 |
| --- | --- |
| 03FBH | カーソル値・実アドレス変換（40字モード）サブルーチン<br>●メモリの FF80H・FF81H の内容を X 軸・Y 軸のカーソル・データとみなして実際の I/O アドレスを計算する。<br>レジスタ：HL，BC，DE レジスタ以外保存<br>出力：BC レジスタに実アドレス |
| 0410H | コントロール・コードの実行<br>●レジスタ：HL，BC，DE，A レジスタ以外保存<br>入力：A レジスタにコントロール・コード |
| 042BH | 画面消去，カーソル Home セット<br>●レジスタ：HL，BC，DE，A レジスタ以外保存 |
| 044DH | カーソル位置セット<br>●レジスタ：全レジスタ保存<br>入力：H に Y 座標<br>L に X 座標 |
| 0451H | CRTC データ設定サブルーチン<br>●0B3FH より14バイト分の CRTC データをセット，音楽を停止させる。 |
| 047EH | サブ CPU 80C49 へキーベクタ値を送る |
| 0523H | サブ CPU 80C49 からデータ受信<br>●レジスタ：A レジスタ以外保存<br>出力：A レジスタに受信データ |
| 052EH | サブ CPU 80C49 へデータを送信<br>●レジスタ：全レジスタ保存<br>入力：A レジスタに送信データ |
| 054FH | “ IPL is loading ” を表示して以下を実行する。 |
| 0555H | ファイルネームを表示<br>●FF01H から13文字分のファイル・ネームを画面に表示<br>レジスタ：DE，A レジスタ以外保存 |
| 0564H | テープよりデータ部をロード<br>●エラーのとき，エラーメッセージ表示ルーチンへジャンプ（JP 00F5H）<br>レジスタ：D，A 以外保存<br>入力：HL レジスタにロード先アドレス<br>BC レジスタに転送バイト数 |
| 0567H | テープよりインフォメーション部をロード<br>●エラーのとき，エラーメッセージ表示ルーチンへジャンプ（JP 00F5H） |

| アドレス | 内　容 |
|---|---|
| 05F9H | テープのトップマークをさがす<br>●レジスタ：A レジスタ以外保存<br>出力：キャリーフラグが 0 のとき OK，1 のときエラー |
| 0641H | ボーレート設定用時間待ち |
| 0648H | カセット動作制御<br>●レジスタ：全レジスタ保存<br>入力：A レジスタにカセット・コマンド値<br>00H：EJECT<br>01H：STOP<br>02H：PLAY<br>03H：FF<br>04H：REW<br>05H：APSS ＋ 1<br>06H：APSS － 1<br>0AH：REC |
| 0652H | カセット状態センス<br>●レジスタ：A レジスタ以外保存<br>出力：A レジスタのビットで返す（図9-4参照）。 |
| 065AH | バックアップ・タイマーから日，曜日を読む<br>●FF18H から 3 バイトに日，曜日を格納する |
| 0673H | 日，曜日，時間を読み込んで画面に表示<br>●レジスタ：A レジスタ以外保存 |
| 06A3H | “／”を画面に表示<br>●レジスタ：A レジスタ以外保存 |
| 06A8H | HL レジスタが示すメモリの内容を16進 2 桁で表示<br>●HL レジスタの示すメモリの内容を16進 2 桁で表示して<br>HL レジスタの値を 1 増加させる。<br>レジスタ：A，HL 以外保存<br>入力：HL に表示させるメモリのアドレス |
| 06ADH | 06A8H のサブルーチンを実行後，“：”を表示 |
| 06B5H | 日，曜を表示 |
| 06E9H | A レジスタの値を16進 2 桁で画面に表示<br>●レジスタ：A レジスタ以外保存<br>入力：A レジスタに表示させるデータ |

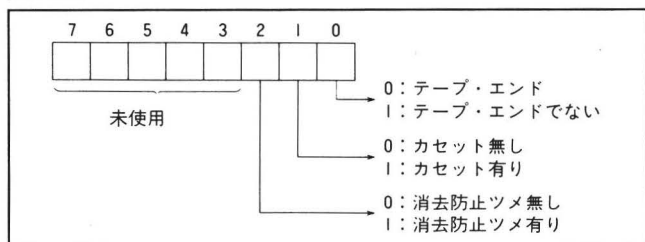
図9 4 **カセットの状態センス**

0700H～0B3DH まではテレビタイマーセット用のプログラムが入っています。この中には，他のプログラムで利用価値のある汎用サブルーチンはあまりみつかりませんでした。よって，0700H～0B3DH 全体を一つのサブルーチンとして利用するのが適切だと思われます。

表9 3 **テレビタイマーセット用サブルーチン**

| アドレス | 内　容 |
|---|---|
| 0700H | タイマーセット・プログラム・スタート<br>●画面消去：タイマーセット画面表示等<br>●ESC キーを押すとリターンする。 |
| 089BH | 文字色を黄色に設定，以下を実行 |
| 089DH～08B6H | A レジスタの値を文字色としてメッセージを16行目に表示。<br>後に文字色を白に戻す。<br>●レジスタ：A，HL，DE 以外保存<br>●入力：A………文字色<br>DE ……文字列先頭アドレス |
| 0820H | カーソルを次の TAB 位置へ移動させる<br>●レジスタ：A，HL 以外保存 |

**表9-4 IPL ROM データ・エリア**

＊各メッセージ・データの終わりは 00H になっている。

| アドレス | 内　容 |
|---|---|
| 0B3EH～0B4AH | CRT 設定データ |
| 0B4BH～0B5CH | タイマーセット画面での X 軸方向の TAB 位置 |
| 0B5DH～0B64H | タイマーセット画面での Y 軸方向の TAB 位置 |
| 0B65H～0B7FH | メッセージ " Push (F, R, C or T) Key ～" |
| 0BA0H～0BB6H | メッセージ " Drive No. ? (0－3) " |
| 0BB7H～0BC7H | メッセージ " IPL is loading " |
| 0BC8H～0BEBH | メッセージ " IPL is looking for a Program from " |
| 0BECH～0BEFH | メッセージ " CMT " |
| 0BF0H～0BF2H | メッセージ " FD " |
| 0BF3H～0C07H | メッセージ " Program load error " |
| 0C08H～0C1FH | メッセージ " Make ready any device " |
| 0C20H～0C34H | メッセージ " File mode error " |
| 0C35H～0C4CH | メッセージ " IPL is under preparing " |
| 0C4BH～0C93H | メッセージ " Please turn on the Power SW. " |
| 0C94H～0CB4H | 2倍角文字 " TV Timer control " |
| 0CB5H～0CDBH | メッセージ " DATA Error ‼ " |
| 0CDCH～0CF7H | TV タイマーセット時メッセージ表示位置テーブル |
| 0D01H～0D27H | メッセージ " Year 00-99 " |
| 0D28H～0D4EH | メッセージ " Month 01-12orXX " |
| 0D4FH～0D75H | メッセージ " Day 01-31 or XX " |
| 0D76H～0D9CH | メッセージ " SUN MON TUE WED THU FRI SAT or XX " |
| 0D9DH～0DC3H | メッセージ " Hour 00-23 or XX " |
| 0DC4H～0DEAH | メッセージ " Minute 00-59 " |
| 0DEBH～0E11H | メッセージ " Second 00-59 " |
| 0E12H～0E38H | メッセージ " TV Power ON ? ( Y or N ) " |
| 0E39H～0E5FH | メッセージ " TV Channele 1-12 " |
| 0E60H～0E86H | スペース38個 |
| 0E87H～0E8AH | " SUN " |
| 0E8BH～0E8EH | " MON " |
| 0E8FH～0E92H | " TUE " |
| 0E93H～0E96H | " WED " |
| 0E97H～0E9AH | " THU " |
| 0E9BH～0E9EH | " FRI " |
| 0E9FH～0EA2H | " SAT " |
| 0EA3H～0EA6H | " XXX " |
| 0EA7H～0EACH | " ON CH " |
| 0EADH～0EB2H | " TIMER " |
| 0EB3H～0EC5H | " XX / XX　XXX　XX : XX　OFF " |
| 0EC6H～0ECEH | " OFF " ＋スペース 5 個 |
| 0ECFH～0EF0H | "[ ESC ] ＝ Exit, [ CLR ] ＝……" |

表9 5 ワーク・エリア

| アドレス | 内　容 |
| --- | --- |
| FE00 | ディスク・ロード用バッファ |
| FF00～FF1F | インフォメーション・バッファ（図9-3参照） |
| FF20～FF25 | タイマー用バッファ |
| FF80 | カーソル X 座標 |
| FF81 | カーソル Y 座標 |
| FF85 | 入力キーデータ |
| FF86 | 文字色データ |
| FF87 | ログイン・ドライブ No. |
| FF88, FF89 | スタック・ポインタ退避用ワーク |
| FF8A, FF8B | ディスク・エラージャンプ・アドレス |
| FF8C～ | スタック用ワーク |

# 9-2-1 IPL ROMの活用例

IPL ROMの活用例としては，小規模なユーティリティなどを自作するときに1文字キー入力やメッセージ表示などの汎用サブルーチンを利用することがあげられます。そうすることによってメイン・プログラムは極力小さくできます。

また，64Kバイトのメイン・メモリをできるだけ多く使用したいプログラムでの活用は特に有効です。具体的には，テープのバックアップをとるためのプログラムが良い例です。60Kバイト近い大きなプログラムのバックアップをとりたいときはグラフィックRAMを利用する方法もありますが，IPL ROMを活用すればより簡単にできます。なぜなら，IPL ROM内にテープの読み込みサブルーチンが存在するからです。あとはテープへの書き込み部分とメイン・プログラムを作成すれば良いわけなので残りの膨大な容量をバッファとして使えます。

バックアップ・プログラムはカセットの章にゆずるとして，ここではIPL ROMを活用した小さなユーティリティを紹介します。

テープにセーブされたマシン語プログラムを解析するとき，まず知っておきたいデータがあります。プログラムの先頭アドレスや実行アドレスです。**リスト9-1**はテープを読んで，これらのインフォメーション・データを画面に表示するプログラムです。

何をセーブしたのかわからなくなってテープを調べたり，プログラムの整理をするのにも役立つでしょう。

このプログラムはIOCSもモニタも不用で，これだけ単独で動きます。

リスト9 1　　No. 1

```
 1:
 2:                      ;
 3:                      ;       File Information Display
 4:                      ;                                      LIST 9-1
 5:                      ;
 6:   C000                       ORG     0C000H
 7:                      ;
 8:                      ; --  IPL ROM ENTRY  --
 9:                      ;
10:   042B =             CLS     EQU     042BH
11:   03CB =             MSG     EQU     03CBH
12:   0355 =             INPUT   EQU     0355H
13:   0567 =             INFRED  EQU     0567H
14:   0555 =             FNAME   EQU     0555H
15:   03D9 =             ACCPRT  EQU     03D9H
16:   0648 =             CMT     EQU     0648H
17:   06E9 =             APRT    EQU     06E9H
18:                      ;
19:   C000 061D                  LD      B,1DH           ;IPL ROM Connect
20:   C002 ED79                  OUT     (C),A
21:   C004 CD2B04                CALL    CLS             ;CLS
22:   C007 119EC0        P1:     LD      DE,MSG0
23:   C00A CDCB03                CALL    MSG
24:   C00D CD5503        LOOP1:  CALL    INPUT
25:   C010 FE0D                  CP      0DH
26:   C012 20F9                  JR      NZ,LOOP1
27:   C014 CDD903                CALL    ACCPRT
28:   C017 2100FF                LD      HL,0FF00H
29:   C01A 012000                LD      BC,32
30:   C01D CD6705                CALL    INFRED
31:   C020 20E5                  JR      NZ,P1
32:   C022 3E01                  LD      A,01H           ;CMT Stop
33:   C024 CD4806                CALL    CMT
34:                      ;
35:   C027 11C0C0                LD      DE,MSG1         ;File Name
36:   C02A CDCB03                CALL    MSG
37:   C02D CD5505                CALL    FNAME
38:   C030 3E0D                  LD      A,0DH
39:   C032 CDD903                CALL    ACCPRT
40:                      ;
41:   C035 11D4C0                LD      DE,MSG2         ;Start Address
42:   C038 2A14FF                LD      HL,(0FF14H)
43:   C03B CD8EC0                CALL    ADPRT
44:                      ;
45:   C03E 2A14FF                LD      HL,(0FF14H)     ;End   Address
46:   C041 EB                    EX      DE,HL
47:   C042 2A12FF                LD      HL,(0FF12H)
48:   C045 19                    ADD     HL,DE
49:   C046 2B                    DEC     HL
50:   C047 11E7C0                LD      DE,MSG3
51:   C04A CD8EC0                CALL    ADPRT
52:                      ;
53:   C04D 11F9C0                LD      DE,MSG4         ;Exec  Address
54:   C050 2A16FF                LD      HL,(0FF16H)
55:   C053 CD8EC0                CALL    ADPRT
56:                      ;
57:   C056 110BC1                LD      DE,MSG5
58:   C059 CDCB03                CALL    MSG
59:   C05C 3A00FF                LD      A,(0FF00H)      ;File Kinds
```

No. 2

```
 60:  C05F 1130C1          LD     DE,MSG51
 61:  C062 FE01            CP     01H
 62:  C064 2811            JR     Z,P2
 63:  C066 1141C1          LD     DE,MSG52
 64:  C069 FE02            CP     02H
 65:  C06B 280A            JR     Z,P2
 66:  C06D 114FC1          LD     DE,MSG53
 67:  C070 FE04            CP     04H
 68:  C072 2803            JR     Z,P2
 69:  C074 115AC1          LD     DE,MSG54
 70:  C077 CDCB03   P2:    CALL   MSG
 71:                ;
 72:  C07A 111EC1          LD     DE,MSG6         ;Pass Word
 73:  C07D CDCB03          CALL   MSG
 74:  C080 3A11FF          LD     A,(0FF11H)
 75:  C083 CDE906          CALL   APRT
 76:  C086 3E48            LD     A,'H'
 77:  C088 CDD903          CALL   ACCPRT
 78:  C08B C307C0          JP     P1
 79:                ;
 80:  C08E CDCB03   ADPRT: CALL   MSG
 81:  C091 7C              LD     A,H
 82:  C092 CDE906          CALL   APRT
 83:  C095 7D              LD     A,L
 84:  C096 CDE906          CALL   APRT
 85:  C099 3E48            LD     A,'H'
 86:  C09B C3D903          JP     ACCPRT
 87:                ;
 88:  C09E 0D0D0D0D MSG0:  DEFB   0DH,0DH,0DH,0DH,0DH
 89:  C0A3 54617065        DEFM   'Tape Set and Hit RETURN Key '
 90:  C0BF 00              DEFB   0
 91:  C0C0 0D0D0D   MSG1:  DEFB   0DH,0DH,0DH
 92:  C0C3 46696C65        DEFM   'File Name     = '
 93:  C0D3 00              DEFB   0
 94:  C0D4 0D0D     MSG2:  DEFB   0DH,0DH
 95:  C0D6 53746172        DEFM   'Start Address = '
 96:  C0E6 00              DEFB   0
 97:  C0E7 0D       MSG3:  DEFB   0DH
 98:  C0E8 456E6420        DEFM   'End   Address = '
 99:  C0F8 00              DEFB   0
100:  C0F9 0D       MSG4:  DEFB   0DH
101:  C0FA 45786563        DEFM   'Exec  Address = '
102:  C10A 00              DEFB   0
103:  C10B 0D0D     MSG5:  DEFB   0DH,0DH
104:  C10D 46696C65        DEFM   'File Kinds    = '
105:  C11D 00              DEFB   0
106:  C11E 0D       MSG6:  DEFB   0DH
107:  C11F 50617373        DEFM   'Pass Word     = '
108:  C12F 00              DEFB   0
109:                ;
110:  C130 4D414348 MSG51: DEFM   'MACHINE LANGUAGE'
111:  C140 00              DEFB   0
112:  C141 42415349 MSG52: DEFM   'BASIC PROGRAM'
113:  C14E 00              DEFB   0
114:  C14F 41534349 MSG53: DEFM   'ASCII FILE'
115:  C159 00              DEFB   0
116:  C15A 3F3F3F   MSG54: DEFM   '???'
117:  C15D 00              DEFB   0
118:                ;
119:  C15E                 END
```

IPL ROMのテープ読み込みサブルーチンを利用するにあたって，1つだけ難点があります。それは，ある種のエラーが生ずると，エラーメッセージ表示ルーチンへジャンプし，スタック・ポイントもクリアしてしまい，メイン・ルーチンへ戻れなくなる点です。

このサンプル・プログラムでもエラー対策を行っていないので，少々使いづらくなっています。

対処のしかたとしては，IPL ROMの内容をメイン・メモリに転送し，エラー処理ルーチンを書きかえてメイン・メモリ上のプログラムを使用する方法が考えられます。IPL ROMが接続されていてもデータの書き込みはメイン・メモリの方へ行なわれるので簡単に転送できます。

テープのエラー処理ルーチンは00EBH～010DHにあります。00FDHあたりにJP命令を入れると良いでしょう。

IPL ROMをメイン・メモリに転送して使用するときはもう数ヵ所変更しなければなりません。それは，ROMの場合，1命令実行するたびに1サイクル余分に時間がかかりますが，RAM上で実行するときはそれがないので，テープ読み込みなどのタイミングが合わなくなってしまいます。ですから，各タイミングに使われているデータ値を変えなければなりません。0642 Hにテープ用ディレイ・カウンタの値が入っているので適切な値に変えてください。

# APPENDIX

1. IOCS
2. IOCS 内ワーク・エリア
3. I/O ポート
4. サブ CPU (80C49) コマンド表

# 付録1 IOCS

キーボード入力

| ラベル名 | アドレス | 内　容 |
| --- | --- | --- |
| INPUTF | 0 0 0 3 | ●機能　1行入力<br>●入力　DE……データ入力アドレス<br>●出力　DE……入力データ先頭アドレス<br>キャリークラブ…0のとき，リターン・および，CTRL＋Jによる正常入力。<br>1のとき，BREAK・キーおよび，CTRL＋Dによるキャンセル。この場合，Aレジスタの値を見て，BREAKかCTRL＋Dかを判断する。<br>A　……キャリーフラグが1のときのみ意味を持ち，BREAKなら03H，CTRL＋Dなら04Hがセットされる。<br>●レジスタ　AF以外<br>●説明　1行分スクリーン・エディットを行ないDEレジスタで指定されたアドレスから1行分のデータを取り込む。このサブルーチンから戻るには次の4種類のキー入力による。<br>①リターン・キーおよびCTRL＋M<br>……正常入力<br>② CTRL＋J<br>……現在のカーソルより前のデータの正常入力<br>③ SHIET＋BREAKおよびCTRL＋C<br>入力キャンセル<br>④ CTRL＋D<br>…入力キャンセル<br>③，④の入力キャンセルでリターンした場合は，DEレジスタの内容は変化しない。 |

| ラベル名 | アドレス | 内　容 |
|---|---|---|
| BINPUT | 015A | ●機能　1行入力<br>●入力　DE…データ入力アドレス<br>●出力　DE……入力データ先頭アドレス<br>キャリーフラグ…0のとき，リターン・キーおよび，CTRL＋Jによる正常入力。<br>1のとき，BREAK・キーおよび，CTRL＋Dによる入力キャンセル。<br>A………キャリーフラグが1のときのみ意味を持ち，BREAKなら03H，CTRL＋Dなら04Hがセットされる。<br>●レジスタ AF，DE以外保存<br>●説明　INPUTFのサブルーチンと同様1行入力のサブルーチンだが，BASICのINPUT文用に用意されたものである。入力開始桁よりも前にあるメッセージ(プロンプト)は入力しない（入力開始桁までDEレジスタの値を進める)。 |
| INKEY$ | 001B | ●機能　1文字入力<br>●入力　A………INKEY$のモード値<br>●出力　A……… 1文字入力したキーのASCIIコード<br>●レジスタ　AF以外<br>●説明　サブルーチンを呼ぶ前にAレジスタにモードをセットする。その値により返ってきたときのAレジスタの意味や途中の動作が違う。<br>モード値<br>● FFHのとき…リアルタイム・キー入力。<br>新しいキーが押されたときや、リピート・モードでリピートがONになるごとに押されているキーのASCIIコードを返す。キーが押されていないときは0を返す。<br>● 01Hのとき……カーソル待ちで1文字入力。<br>キーが押されるまでループ。押されたキーのASCIIコードを返す。リピード・モードではリピートしたキーのコードを返す。<br>● 00Hのとき……サブCPUからの ASCII コード部8ビット・データをそのまま返す。<br>● 02Hのとき……サブCPUからのファンクション・コード部8ビットをそのまま返す。図A-1参照。 |

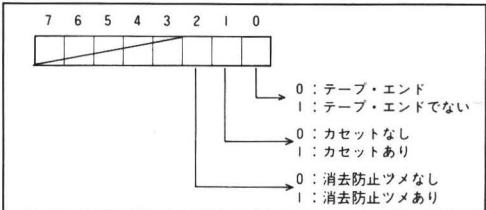
| ラベル名 | アドレス | 内　　容 |
| --- | --- | --- |
| BRKCHK | 0 0 4 A | ●機能　SHIFT＋BREAK が押されたかの判断<br>●出力　ゼロフラグ…1のとき，SHIFT＋BREAK が押された。0のとき，押されていない。<br>●レジスタ　AF以外保存 |

画面，プリンタへの出力

| ラベル名 | アドレス | 内　　容 |
| --- | --- | --- |
| PRINT | 0 0 0 B | ●機能　文字列のプリント<br>●入力　DE……文字列の先頭アドレス<br>●レジスタ　AF以外保存<br>●説明　DEで示すアドレスから始まる文字列を画面に出力する。文字列データは ASCII コードであり，エンド・コードは 00H である。<br>01H〜1FH のコントロール・コードはその処理が実行される。 |
| ACCPRT | 0 0 1 3 | ●機能　1文字出力<br>●入力　A………出力する ASCII コード<br>●レジスタ　すべて保存<br>●説明　Aレジスタにある ASCII コードの文字（20H〜FFH）を画面に表示する。コントロール・コード（00H〜1FH）はその処理が実行される。 |
| ACCDIS | 0 4 C 8 | ●機能　1文字出力<br>●入力　A………出力する ASCII コード<br>●レジスタ　すべて保存<br>●説明　ACCPRT と同様の1文字出力のサブルーチンだが，コントロール・コード（00H〜1FH）も画面に表示して，コントロール・コードとしての処理はしない。 |
| CTRLJB | 0 5 7 7 | ●機能　コントロール・コード処理<br>●入力　A………コントロール・コード（00H〜1FH）<br>●レジスタ　AF，BC，DE，HL 以外は保存<br>●説明　Aレジスタで指定されたコントロール・コードの処理を行う。ACCPRT で 00H〜1FH を出力したのと同じ。 |

| ラベル名 | アドレス | 内　容 |
|---|---|---|
| SPPRT | 0 4 B A | ●機能　スペースを1個出力<br>●レジスタ　AF 以外保存<br>●説明　スペース（ASCII コード 20H）を画面に出力。 |
| TABPRT | 0 4 A B | ●機能　10文字間隔の TAB 処理<br>●レジスタ　AF 以外保存<br>●説明　X 座標が 10の倍数になるまでスペースを出力。 |
| CR1 | 0 4 A 7 | ●機能　改行<br>●レジスタ　AF 以外保存<br>●説明　次の行の先頭へカーソルを移動する。 |
| CR2 | 0 4 A 3 | ●機能　行の先頭でなければ改行<br>●レジスタ　AF 以外保存<br>●説明　現在のカーソル位置が行の先頭でないならば改行し，行の先頭なら改行しない。 |
| ACCLPL | 1 2 D C | ●機能　プリンタへの1文字出力<br>●入力　A……出力する ASCII コード<br>●レジスタ　F 以外保存<br>●説明　A レジスタにある ASCII コードをプリンタに出力する。 |
| CR1LPL | 1 2 D 5 | ●機能　プリンタへの改行コード出力<br>●レジスタ　AF 以外保存<br>●説明　改行（LF＝0AH）をプリンタに出力する。 |
| TABLPL | 1 3 1 5 | ●機能　プリンタへの HTAB 出力<br>●レジスタ　AF 以外保存<br>●説明　水平タブ（HTAB）をプリンタに出力する。 |
| ACCPRP | 1 4 2 0 | ●機能　画面またはプリンタへの1文字出力<br>●入力　A ……出力するASCII コード<br>●レジスタ　F,AF' 以外保存<br>●説明　FILOUT（1472H）が0のとき画面に出力，1のときプリンタに出力。FILOUT の内容はモニタのＰコマンドで切り換えることができる。 |

| ラベル名 | アドレス | 内　容 |
| --- | --- | --- |
| PRINTP | 1 4 2 F | ●機能　画面またはプリンタに文字列出力<br>●入力　DE……文字列の先頭アドレス<br>●レジスタ　AF, AF'以外保存<br>●説明　FILOUT(1472H)が0のとき画面に出力，1のときプリンタに出力。 |
| TABPRP | 1 4 3 C | ●機能　画面またはプリンタにHTAB出力<br>●レジスタ　AF以外保存<br>●説明　水平タブを，FILOUT(1472H)が0のとき画面に，1のときプリンタに出力する。 |
| CR1PRP | 1 4 4 6 | ●機能　画面またはプリンタに改行コード出力<br>●レジスタ　AF以外保存<br>●説明　改行コードを，FILOUT（1472H）が0のとき画面に，1のときプリンタに出力する。 |

## 表示のコントロール

| ラベル名 | アドレス | 内　容 |
| --- | --- | --- |
| WIDTH80 | 0 9 8 C | ●機能　WIDTH　80<br>●レジスタ　AF, BC, DE, HL以外は保存<br>●説明　80字のモード指定をするとともにIOCSのワーク(WIDTH0：0007H)に50Hがセットされる。この際，画面はテキスト，グラフィック共にクリアされ，コンソールは解除され，最大値となる。<br>ただし，SCRMOD(0A8BH)が02Hならばグラフィック画面はクリアされない。 |
| WIDTH40 | 0 9 9 8 | ●機能　WIDTH　40<br>●レジスタ　AF, BC, DE, HL以外は保存<br>●説明　40字のモード指定をするとともにIOCSのワーク(WIDTH0：0007H)に28Hがセットされる。この際，画面はテキスト，グラフィック共にクリアされ，コンソールは解除され，最大値となり，スクリーンも0ページ入出力のモードとなる。<br>ただし，SCRMOD（0A8B）が02Hならばグラフィック画面はクリアされない。 |

| ラベル名 | アドレス | 内容 |
|---|---|---|
| SCRNOT | 0 9 C 0 | ●機能　表示ページ指定<br>●入力　A……… 0のとき，ページ0指定<br>　　　　　　　1のとき，ページ1指定<br>●レジスタ　AF以外保存<br>●説明　Aレジスタで示すページを出力とする。このルーチンは，WIDTH40のモードのとき動作し，WIDTH 80のモードのときは何もしない。 |
| SCRNIN | 0 9 F 5 | ●機能　書き込み画面のページ指定<br>●入力　A……… 0のとき，ページ0指定<br>　　　　　　　1のとき，ページ1指定<br>●レジスタ　AF以外保存<br>●説明　Aレジスタで示すページを書き込み対象の画面ページとする。<br>このルーチンはWIDTH40のモードのみ有効。 |
| CTRLD? | 0 A 3 F | ●機能　コンソールと入出力ページの初期化<br>●レジスタ　AF以外保存<br>●説明　コンソール（テキスト画面の座標範囲）を最大値に戻し，入出力ページを0ページに指定する。 |
| STPRIO | 0 A 5 A | ●機能　パレット，プライオリティのセット<br>●レジスタ　AF，BC，D，HL以外保存<br>●説明　RPRIOF(00F6H)の青のパレット，GPRIOF(00F7H)赤のパレット，BPRIOF（00F8H）緑のパレット，TPRIOF(00F9H)プライオリティの各ワークの値をI/0アドレスのパレット青（1000H)，パレット赤(1100H)，パレット緑（1200H)，プライオリティ(1300H）にセットする。 |
| CLST | 0 A 6 B | ●機能　テキスト・クリア<br>●レジスタ　AF，BC，D，HL以外保存<br>●説明　テキスト画面をクリアする。このとき画面に埋められる文字はCLSCHR（0027H）に，アトリビュートはCOLORF（0026H）にストアされている。 |

| ラベル名 | アドレス | 内　容 |
| --- | --- | --- |
| CLSG | 0A8F | ●機能　グラフィック・クリア<br>●レジスタ　AF, BC以外保存<br>●説明　グラフィック画面を同時アクセス・モードでクリアする。 |
| ADRCAL | 054A | ●機能　カーソル位置のアドレス取り出し<br>●出力　HL……テキスト VRAM アドレス<br>●レジスタ　AF, BC, HL 以外は保存<br>●説明　現在のカーソル位置のテキスト VRAM アドレスを HL レジスタに返す。アトリビュート・アドレスは, HL－1000H の位置である。 |
| ADRCA2 | 054D | ●機能　テキスト VRAM のアドレス計算<br>●入力　L………テキスト X 座標の値<br>H………テキスト Y 座標の値<br>●出力　HL……テキスト VRAM アドレス<br>●レジスタ　AF, BC, HL 以外保存<br>●説明　HL レジスタで指定したテキスト座標に対応するテキスト VRAM アドレスを計算し, HL レジスタに返す。アトリビュート・アドレスは, HL－1000H の値である。 |
| SAVE1 | 003B | ●機能　ファイル・コントロールブロック (FCB) をカセットテープにセーブ<br>●入力　HL……FCB の先頭アドバイス<br>BC……FCB のバイト数(X1では0020Hを指定する)<br>●出力　A………0のとき, セーブ OK<br>1のとき, 途中で BREAK<br>3のとき, テープがセットされていない。<br>4のとき, WRITE PROTECT 状態である。<br>5のとき, その他 (テープ・エンドおよびカセット操作キーが押された)。<br>キャリーフラグ……0のとき, OK<br>1のとき, ERROR<br>●レジスタ　AF 以外保存<br>●説明　ファイル・コントロールブロック (FCB) をカセットにセーブする。図 A-2 参照。 |

| ラベル名 | アドレス | 内　　容 |
| --- | --- | --- |
| SAVE2 | 0 0 3 E | ●機能　データをカセットテープにセーブ<br>●入力　HL……セーブするデータの先頭アドレス<br>BC……セーブするデータのバイト数<br>●出力　A………0のとき，セーブOK<br>1のとき，途中でBREAK<br>3のとき，テープがセットされていない<br>4のとき，WRITE PROTECT状態である<br>5のとき，その他。<br>キャリーフラグ…0のとき，OK<br>1のとき，ERROR<br>●レジスタ　AF以外保存<br>●説明　HLで示されるメイン・メモリのアドレスからBCバイト分をカセットにしセーブする。BC=0のときは65536バイトセーブされる |
| LOAD1 | 0 0 4 1 | ●機能　ファイル・コントロールブロック（FCB）をカセットテープからロードする。<br>●入力　HL……FCBをロードする先頭アドレス<br>BC……FCBのバイト数（X1では0020H)<br>●出力　A………0のとき，ロードOK<br>1のとき，途中でBREAK<br>2のとき，CHECK SUM ERROR<br>3のとき，テープがセットされていない。<br>5のとき，その他<br>キャリーフラグ…0のとき，OK<br>1のとき，ERROR<br>●レジスタ　AF以外保存<br>●説明　HLで示されるメイン・メモリのアドレスにカセットテープからファイル・コントロールブロック（FCB）をロードする。 |

| ラベル名 | アドレス | 内　容 |
| --- | --- | --- |
| LOAD2 | 0 0 4 4 | ●機能　データをカセットテープからロードする。<br>●入力　HL……ロードする先頭アドレス<br>BC……ロードするバイト数<br>●出力　A………0のとき，ロードOK<br>1のとき，途中でBREAK<br>2のとき，CHECK SUM ERROR<br>3のとき，テープがセットされていない<br>5のとき，その他<br>キャリーフラグ…0のとき，ロードOK<br>1のとき，ERROR<br>●レジスタ　AF以外保存<br>●説明　HLで示されるメイン・メモリのアドレスからBCバイト分のデータを，カセットテープよりロードする。BC＝0ならば65536バイト分ロードされる。 |
| VERFY2 | 0 0 4 7 | ●機能　カセットテープのデータとメイン・メモリのデータを比較。<br>●入力　HL……比較するメイン・メモリ先頭アレス。<br>BC……比較するバイト数<br>●出力　A………0のとき，比較OK（同じデータである。）<br>1のとき，途中でBREAK<br>2のとき，CHECK SUM ERROR および VERIFY ERROR（データが異なっている。）<br>3のとき，テープがセットされていない。<br>5のとき，その他<br>キャリーフラグ…0のとき，OK<br>1のとき，ERROR<br>●レジスタ　AF以外保存<br>●説明　HLで示されるメイン・メモリのアドレスからBCバイト分のデータを，カセットテープのデータと比較チェックする。もし，内容が違っていればAレジスタに02Hを返す。 |

| ラベル名 | アドレス | 内　　容 |
|---|---|---|
| CMTCOM | 0DEC | ●機能　カセットに対してコマンドを実行する。<br>●入力　A………カセット・コマンド値<br>●レジスタ　AF 以外保存<br>●説明　A レジスタのコマンド値によって，以下のような動作を行う。<br>00H……EJECT<br>01H……STOP<br>02H……PLAY<br>03H……FF（FAST）<br>04H……REW<br>05H……APSS FF（APSS＋1）<br>06H……ASPP REW（APSS－1）<br>0AH……REC |
| CMTSNS | 0DF6 | ●機能　カセットの状態を示す。<br>●出力　A……カセットの状態<br>●レジスタ　AF 以外保存<br>●説明　カセットの状態を A レジスタの各ビットで返す。各ビットの意味は下図の通り。<br>**カセットの状態**<br>ビット 7 6 5 4 3 2 1 0<br>ビット0：0：テープ・エンド　1：テープ・エンドでない<br>ビット1：0：カセットなし　1：カセットあり<br>ビット2：0：消去防止ツメなし　1：消去防止ツメあり |
| FMPRHL | 1321 | ●機能　FCB のファイル・ネームをプリントする。<br>●入力　DE……メッセージ出力先頭<br>HL……FCB 先頭アドレス<br>●レジスタ　AF，D 以外保存<br>●説明　DE から始まるメッセージをプリントした後に HL で示す FCB 中のファイル・ネームをプリントする。 |

| ラベル名 | アドレス | 内　　容 |
|---|---|---|
| FNMTCH | 1 3 4 E | ●機能　ファイル・ネームおよびパスワードの比較<br>●入力　HL……FCB の先頭アドレス<br>●出力　セロ・フラグ…1のとき，一致<br>　　　　　　　　　　0のとき，不一致<br>●レジスタ　AF 以外保存<br>●説明　HLで示すFCBと DIRIMG（1480H）で示す FCB のファイル・ネームおよびパスワードを比較する。 |
| SETDI? | 1 3 9 4 | ●機能　FCB にファイル・ネームをセット<br>●入力　DE……ファイル・ネームが格納されているバッファの先頭アドレス<br>●レジスタ　AF，BC，DE，HL 以外保存<br>●説明　DE で示されるバッファにあるファイル・ネームとパスワードを DIRIMG（1480H）のFCBにセットする。また，同時に日付けも FCB にセットする。 |

PSG制御

| ラベル名 | アドレス | 内　　容 |
|---|---|---|
| PSGINT | 0 1 3 C | ●機能　PSG の出力を止める<br>●レジスタ　AF，BC，DE 以外保存<br>●説明　PSG の出力を止め，PSG のレジスタ$R_7$～$R_{10}$に次の値をセットする。<br>$R_7$ ← 3FH<br>$R_8$ ← 00H<br>$R_9$ ← 00H<br>$R_{10}$ ← 00H |
| BEEP | 0 7 F 7 | ●機能　ベル音を鳴らす。<br>●レジスタ　AF，BC，D 以外は保存<br>●説明　PSG を使ってベル音を鳴らす。 |

PCG制御

| ラベル名 | アドレス | 内　　容 |
| --- | --- | --- |
| CGSET | 0 0 2 B | ●機能　PCG にデータ・セット<br>●入力　D……… ASCII コード<br>E………セット I/O アドレス上位<br>15H…PCG 青<br>16H…PCG 赤<br>17H…PCG 緑<br>HL……セットするデータの先頭アドレス<br>●出力　HL……先頭アドレス+8 バイト<br>●レジスタ　AF, E, HL 以外保存<br>●説明　PCG にパターンを定義する。 |
| CGREAD | 0 0 3 3 | ●機能　PCG や CG からデータを読み込む<br>●入力　D……… ASCII コード<br>E………リード I/O アドレス上位<br>14H…CG ROM<br>15H…PCG 青<br>16H…PCG 赤<br>17H…PCG 緑<br>HL……データ用バッファ先頭アドレス<br>●出力　HL……先頭アドレス＋ 8 バイト<br>●レジスタ　AF, E, HL 以外保存<br>●説明　キャラクタ・ジェネレータ（CG ROM および PCG RAM）の内容を読み込み，バッファに格納する。 |

サブCPUとの通信

| ラベル名 | アドレス | 内容 |
| --- | --- | --- |
| TRANS49 | 0 B 5 4 | ●機能　サブ CPU80C49 に送信要求コードを送る。<br>●入力　A………80C49 に送る要求コード<br>●レジスタ　AF 以外保存<br>●説明　Z80 から 80C49 に次のようなコントロールを行わせる。<br>1．キーコード処理<br>2．モニタ画面モードの処理<br>3．TV の各種コントロール<br>4．カセットのコントロール<br>5．時刻の設定，読み出し<br>6．タイマーの設定，読み出し<br>送信要求コードは付録 3 を参照。 |
| RECV49 | 0 B 4 9 | ●機能　サブ CPU80C49 からデータを受信。<br>●出力　A……… 80C49から受信したデータ<br>●レジスタ　AF 以外保存<br>●説明　80C49 からデータを受信する。このサブルーチンは TRANS49 と共に，80C49 との通信に用いる。 |

# 付録2 IOCS内ワーク・エリア

## ワーク・エリア

| ラベル名 | アドレス | バイト数（10進） | 内　　容 |
|---|---|---|---|
| LINLIM | 0006 | 1 | ●意味　1行の入力文字数の最大<br>●値　1〜255<br>●説明　INPUTF ,BINPUTなどのサブルーチンで1行に入力できる文字数の最大値。この値以上はバッファに格納されない。 |
| WIDTH0 | 0007 | 1 | （リードのみ）<br>●意味　WIDTHの設定値<br>●値　40または80<br>●説明　現在のスクリーンがWIDTH 40かWIDTH80かを記憶している。40（28H）か80（50H）の値を持つ。 |
| CURX | 000E | 1 | ●意味　カーソルのX座標<br>●値　0〜39（WIDTH40）<br>0〜79（WIDTH80）<br>●説明　現在のカーソル位置のX座標を示す。書きかえることでカーソル位置を変更でる。 |
| CURY | 000F | 1 | ●意味　カーソルのY座標<br>●値　0〜24<br>●説明　現在のカーソル位置のY座標を示す。書きかえることでカーソル位置を変更できる。 |

| ラベル名 | アドレス | バイト数（10進） | 内　容 |
|---|---|---|---|
| CURYST | 0 0 1 6 | 1 | ●意味　テキスト表示エリアのY座標の先頭<br>●値　　0～24<br>●説明　CONSOLE命令でのY方向のスタート座標値。下図参照。<br><br>**CURYST(YS), CURYED(YE), CURXST(XS), CURXED(XE) の関係**<br>(0, 0)<br>( XS, YS )<br>テキスト表示エリア<br>( XE, YE )<br>(39, 24)<br>注）WIDTH40 の場合 |
| CURYED | 0 0 1 7 | 1 | ●意　　テキスト表示エリアのY座標の終わり<br>●値　　(CURYST) ～24<br>●説明　CONSOLE命令でのY方向のエンド座標を示す。 |
| CURXST | 0 0 1 E | 1 | ●意味　テキスト表示エリアのX座標の先頭<br>●値　　0～39　または0～79<br>●説明　CONSOLE命令のX方向のスタート座標を示す。 |
| CURXED | 0 0 1 F | 1 | ●意味　テキスト表示のX座標の終わり。<br>●値　(CURXST) ～39　または<br>(CURXST) ～79<br>●説明　CONSOLE命令のX方向のエンド座標を示す。 |

| ラベル名 | アドレス | バイト数(10進) | 内　容 |
|---|---|---|---|
| COLORF | 0026 | 1 | ●意味　文字のアトリビュート<br>●値　0～255<br>●説明　ACCPRT などのサブルーチンで表示される文字のアトリビュートの値を示す。各ビットの意味は下図の通り。 |
| CLSCHR | 0027 | 1 | ●意味　画面消去用キャラクタ<br>●値　通常 20H（スペース）<br>●説明　文字画面を消すときに埋めつくされるキャラクタの ASCII コードを示す。 |
| KEYDAT | 002E<br>002F | 1<br>1 | (リードのみ)<br>●意味　割り込みキー入力でのキーコード<br>●説明　割り込みによるキー入力で最後に入ってきたキーの値<br>002EH………ASCIIコード<br>002FH……ファクション・コード |
| BRKBUF | 0036 | 1 | (リードのみ)<br>●意味　割り込みキー入力でのBREAKキーバッファ<br>●説明　割り込みによるキー入力で，SHIFT＋BREAK および BREAK が押された場合，03H および 13H が格納されてる。BREAK，一時ストップ処理が終るまでクリアされない。 |

**COLORF ( 0026H ) のビット構成**

| ビット | 説　明 |
|---|---|
| 0<br>1<br>2 | 青<br>赤　COLOR　(0～7)　色指定<br>緑 |
| 3 | CREV　(0/1)　反転文字 |
| 4 | CFLASH (0/1)　点滅文字 |
| 5 | CGEN　(0/1)　CG ROM/RAM |
| 6<br>7 | CSIZE　(0～3)　キャラクタ・サイズ |

| ラベル名 | アドレス | バイト数(10進) | 内　容 |
|---|---|---|---|
| KEYFLG | 0 0 3 7 | 1 | (リードのみ)<br>●意味　有効キー／無効キー判断<br>●説明　割り込みキー入力において有効なキーが入力されたときは0以外，無効なキーが入力されたときと，入力がないときは0となる。 |
| ONKYBF | 0 0 5 1 | 1 | (リードのみ)<br>●意味　ファンクション・キー割り込みフラグ<br>●説明　00Hのときファクション・キーは押されていない。90Hのとき押された。 |
| INTTAB | 0 0 5 2 | 20 | (リードのみ)<br>●意味　割り込みキー処理のシャンプ先アドレス・テーブル<br>●値　先頭2バイト　0346H<br>その他　03D3H<br>●説明　使用されているのは先頭の2バイトのみ。Iレジスタとキーベクトル値の合計がこのINTTABを示している。 |
| NMIAD | 0 0 6 7 | 2 | ●意味　リセット・キーを押したときのコール・アドレス<br>●値　00FAH |
| CONTTB | 0 0 6 9 | 64 | ●意味　コントロール・コード処理のジャンプ先アドレス・テーブル<br>●説明　CTRL＋@，CTRL＋A……の処理先のアドレスが2バイトづつ32組格納されている。 |
| SCRNT0<br>SCRNT1 | 0 0 A 9<br>0 0 C 3 | 26<br>26 | ●意味　テキスト画面の行連続フラグ<br>●説明　テキスト画面のページ0，1のある行が次行とつながっているかどうかの判断，00Hのとき，その行は次行とつながっていない。01Hのときつながっている。 |

<table>
<tr><th>ラベル名</th><th>アドレス</th><th>バイト数<br>(10進)</th><th>内　　容</th></tr>
<tr><td>INICRT</td><td>0 0 D D</td><td>12</td><td>●意味　CRT コントローラ設定データ<br>●説明　CRT コントローラのレジスタ 0 〜12に書き込むデータ。</td></tr>
<tr><td>INIADR</td><td>0 0 E 9<br>0 0 E A</td><td>1<br>1</td><td>(リードのみ)<br>●意味　表示している画面のページ・オフセット値<br>●値　(00E9H)……00Hまたは 04H<br>(00EAH)…………00H<br>●説明　表示している画面のページに応じた I/O アドレスのオフセット値を示す。<br><br>表示画面のオフセット値
<table>
<tr><th>WIDTH</th><th>ページ</th><th>00E9H</th><th>00EAH</th></tr>
<tr><td rowspan="2">40</td><td>0</td><td>00H</td><td>00H</td></tr>
<tr><td>1</td><td>04H</td><td>00H</td></tr>
<tr><td>80</td><td>0</td><td>00H</td><td>00H</td></tr>
</table></td></tr>
<tr><td>INIADW</td><td>0 0 E B<br>0 0 E C</td><td>1<br>1</td><td>(リードのみ)<br>●意味　書き込み画面のページ・オフセット値<br>●値　(00EB)……00H<br>(00EC)……00H または 04H<br>●説明　書き込み画面のページに応じた I/O アドレスのオフセット値を示す。<br><br>アクセス画面のオフセット値
<table>
<tr><th>WIDTH</th><th>ページ</th><th>00EBH</th><th>00ECH</th></tr>
<tr><td rowspan="2">40</td><td>0</td><td>00H</td><td>00H</td></tr>
<tr><td>1</td><td>00H</td><td>04H</td></tr>
<tr><td>80</td><td>0</td><td>00H</td><td>00H</td></tr>
</table></td></tr>
<tr><td>ATREND</td><td>0 0 F 1</td><td>2</td><td>(リードのみ)<br>●意味　アトリビュート VRAM エンド・アドレス＋1<br>●説明　アトリビュートで使用されている最後の VRAM の I/O アドレスに 1 を加えた値。PCG の R/W のときに使われる。</td></tr>
</table>

| ラベル名 | アドレス | バイト数(10進) | 内　容 |
|---|---|---|---|
| ATRLFT | 0 0 F 3 | 1 | (リードのみ)<br>●意味　VRAM 未使用バイト数<br>●説明　アトリビュートおよびテキストの VRAM で，画面表示に使用されていないVRAMのバイト数を示す。 |
| CHREND | 0 0 F 4 | 2 | (リードのみ)<br>●意味　テキスト VRAM エンド・アドレス＋1<br>●説明　テキストで使用されている最後の VRAM の I/O アドレスに 1 を加えた値。 |
| BPRIOF | 0 0 F 6 | 1 | (リードのみ)<br>●意味　青のパレット状態<br>●説明　I/O アドレス 1000H の値をもつワークで，青のパレットの状態を示す。 |
| RPRIOF | 0 0 F 7 | 1 | (リードのみ)<br>●意味　赤のパレット状態<br>●説明　I/O アドレス 1100H の値をもつワークで，赤のパレットの状態を示す。 |
| GPRIOF | 0 0 F 8 | 1 | (リードのみ)<br>●意味　緑のパレット状態<br>●説明　I/O アドレス 1200H の値をもつワークで，緑のパレットの状態を示す。 |
| TPRIOF | 0 0 F 9 | 1 | (リードのみ)<br>●意味　テキストの優先順位<br>●値　初期値　0<br>●説明　I/O アドレス 1300H の値をもつワークで優先順位を示す。 |
| REPTFI | 0 3 6 6 | 1 | ●意味　リピート ON/OFF フラグ<br>●値　0 または 1<br>●説明　0 のとき，キーリピート OFF<br>1 のとき，キーリピート ON |

| ラベル名 | アドレス | バイト数（10進） | 内　　容 |
| --- | --- | --- | --- |
| SCRMOD | 0 A 8 B | 1 | ●意味　グラフィック・クリア判断<br>●値　02H，その他<br>●説明　WIDTH40，WIDTH80でグラフィックをクリアするかしないかの判断用。02Hのときはグラフィックをクリアしない。 |
| CLICKF | 0 E 9 0 | 1 | ●意味　クリック音の制御フラグ。<br>●値　00H，その他<br>●説明　割り込みキー入力のときに入力確認用の音を出すか出さないかのフラグ。0ならクリック音が出る。その他のときは出ない。 |
| KBUFSW | 0 E A 5 | 1 | ●意味　1行入力のとき，先行入力を捨てるかどうかのフラグ<br>●値　00H，その他<br>●説明　値が0のとき先行入力を捨てない。 |
| POINT1 | 0 E A 6 | 1 | ●意味　INBUF内の書き込みポインタ<br>●値　0～3FH<br>●説明　先行入力およびファンクション・キー入力のためのINBUF内の書き込みポインタを示す。 |
| POINT2 | 0 E A 7 | 1 | ●意味　INBUF内の読み出しポインタ<br>●値　0～3FH<br>●説明　先行入力およびファンクション・キー入力のためのINBUF内の読み出しポインタを示す。 |

| ラベル名 | アドレス | バイト数(10進) | 内　容 |
|---|---|---|---|
| INBUF | 0EA8<br>～<br>0EE7 | 64 | ●意味　入力キーデータ・バッファ<br>●説明　先行入力およびファンクション・キー入力のためのデータ・バッファ，POINT1およびPOINT2が，書き込み，読み出しのポインタを表わしている。<br>先行入力ルーチンにより，キーデータがバッファに書き込まれ，それにしたがってPOINT1が進んでいく。このバッファはリング・バッファとなっているので0EE7Hの次は0EA8Hに書き込まれる。<br>POINT1とPOINT2が同じ値になったらバッファが空であることを意味する。 |
| FUNBUF | 0F42<br>～<br>0FE1 | 160 | ●意味　ファンクション・キー定義用ワーク<br>●説明　ファンクション・キーの定義内容が書かれているワーク・エリア。<br>1キーに対して16バイト。よって10×16＝160バイト。ワーク・エリアの構造は，最初の1バイトが定義バイト数(0～15)，残りの15バイトが定義文字。 |

**図A-1 ファンクション・コードのビット構成**

(MSB)　　　　　　　　　　　　　　　　　　　　(LSB)

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | ファンクション | キーデータが 有効　無効 | リピート | GRAPH | CAPS | カナ | SHIFT | CTRL |
| 0 | ●テンキー<br>●ファンクションキー<br>●TVキー<br>●カセット・キー | ●データ・コード(8ビット)が有効である<br>ヌルコード"00"以外が送られてきたとき | ●リピート・データである | ●GRAPHキーが押されている | ●CAPSキーが押されている<br>(LOCKされている) | ●カナキーが押されている<br>(LOCKされている) | ●SHIFTキーが押されている | ●CTRLキーが押されている |
| 1 | ●上記以外 | ●データコード(8ビット)が無効である<br>ヌル・コード"00"が送られてきたとき | ●1回目のデータである | ●GRAPHキーが離されている | ●CAPSキーが離されている | ●カナキーが離されている | ●SHIFTキーが離されている | ●CTRLキーが離されている |

図A 2 ファイル・インフォメーションブロック

# 付録3 I/Oポート

| I/O アドレス | |
|---|---|
| 0000–0FFF | ユーザー I/O ポート |
| 1000–1FFF | システム I/O ポート |
| 2000–27FF | アトリビュート VRAM |
| 3000–37FF | テキスト VRAM |
| 37FF–4000 | 漢字テキストVRAM |
| 4000–7FFF | グラフィック VRAM1 (blue) |
| 8000–BFFF | グラフィック VRAM2 (red) |
| C000–FFFF | グラフィック VRAM3 (green) |

| I/O アドレス | 内　容 | I/O |
|---|---|---|
| 0B00 | 増設 RAM バンク切り換え | OUT |
| 0D** | 外部 RAM アクセス | I/O |
| 0E** | 外部 ROM アクセス | I/O |
| 0F** | フロッピーディスク | I/O |

| I/O アドレス | 内　容 | I/O |
|---|---|---|
| 10** | パレット blue | OUT |
| 11** | パレット red | OUT |
| 12** | パレット green | OUT |
| 13** | プライオリティ | OUT |
| 14** | CG ROM (漢字ROM) | IN |
| 15** | PCG RAM blue | I/O |
| 16** | PCG RAM red | I/O |
| 17** | PCG RAM green | I/O |
| 18*0 | CRTC レジスタ | I/O |
| *1 | データ | |
| 19** | 80C49 (8255①) | I/O |
| 1A*0 | 8255②ポート A | I/O |
| *1 | ポート B | |
| *2 | ポート C | |
| *3 | コントロールレジスタ | |
| 1B** | PSG データ | I/O |
| 1C** | PSG アドレス | OUT |
| 1D** | IPL ROM アクティブ | OUT |
| 1E** | IPL ROM ノンアクティブ | I/O |
| 1F8* | DMA | I/O |
| 90 | SIO チャンネル A データ | I/O |
| 91 | チャンネル A 制御語 | |
| 92 | チャンネル B データ | |
| 93 | チャンネル B 制御語 | |
| A0 | CTC チャンネル 0 | I/O |
| A1 | チャンネル 1 | |
| A2 | チャンネル 2 | |
| A3 | チャンネル 3 | |
| D* | 画面管理 | OUT |
| E* | 黒色制御 | OUT |
| F* | START PORT | IN |

注）アミのかかっている部分は X1Turbo の拡張部分，＊は無効。

# 付録4 サブ CPU (80C49) コマンド表

**サブ CPU コマンド（送信要求コード）**

| 送信要求コード | 後続コード | 内　容 | データの方向 80C49 Z-80A |
|---|---|---|---|
| E3 | キーデータ3バイト | ゲーム用キーデータ読み出し | → |
| E4 | ベクタ値1バイト | キーベクタ値をセット | ← |
| E6 | キーデータ2バイト | キーバッファ読み出し | → |
| E7（TV 制御） | 01 | ボリューム・アップ | ← |
| | 02 | ボリューム・ダウン | |
| | 03 | ボリューム・ノーマル | |
| | 05 | TV 画面 | |
| | 06 | 音声ミュート | |
| | 08 | TV ↔コンピュータ切り換え | |
| | 0A | コントラスト・ノーマル | |
| | 0B | チャンネル・アップ | |
| | 0C | チャンネル・ダウン | |
| | 0D | パワーオフ | |
| | 0E | パワーオン・オフ切りかえ | |
| | 0F | コントラスト・ダウン | |
| | 10 〜 1B | チャンネル1 〜 チャンネル12 | |
| | 1C | TV 画面 | |
| | 1D | コンピュータ画面 | |
| | 1E | スーパーインポーズ（コントラストダウン） | |
| | 1F | スーパーインポーズ（コントラストノーマル） | |
| | 80 | パワーオン | |
| E8 | データ1バイト | TV 送信コード読み出し | → |
| E9（カセット制御） | 00 | EJECT | ← |
| | 01 | STOP | |
| | 02 | PLAY | |
| | 03 | FF | |
| | 04 | REW | |
| | 05 | APSS FF | |
| | 06 | APSS REW | |
| | 0A | REC | |
| EA | データ1バイト | カセット状態読み出し | → |
| EB | データ1バイト | カセットセンサー読み出し | → |
| EC | 日付けデータ3バイト | 日付けセット | ← |
| ED | 日付けデータ3バイト | 日付け読み出し | → |
| EE | 時刻データ3バイト | 時刻セット | ← |
| EF | 時刻データ3バイト | 時刻読み出し | → |
| D0 〜 D7 | タイマーデータ6バイト | タイマー0 〜 タイマー7 にデータセット | ← |
| D8 〜 DF | タイマーデータ6バイト | タイマー0 〜 タイマー7 からデータ読み出し | → |

注）アミのかかっている部分は X1 Turbo 用。

## ■ 参考文献 ■


⑴ 大川　善邦　著
『演算プログラムの作り方』産報出版社

⑵ マイコンピュータ No.2
『インターフェース LSI の研究』

⑶ マイコンピュータ No.8
『続・インターフェース LSI の研究』

⑷ 清水　保弘　著
『X1マシン語活用百科』産業報知センター

⑸ 渡部　茂　他共著
『マイコン徹底研究』日本経済新聞

⑹ 『CZ-800P 取扱説明書』㈱シャープ

⑺ 『RP-80 II取扱説明書』㈱エプソン

⑻ 『PIO-3055』㈱I・Oデータ機器

⑼ 『MZ-80B　オーナーズマニュアル』㈱シャープ

⑽ 『CZ-800C BASIC MANUAL』㈱シャープ

// ReSharper disable once

# X1リファレンスノウト

昭和60年4月5日　初版発行　　定価 2,500円

著　者　杉浦勇一，難波生，仲谷和人，松村守
発行人　塚本慶一郎
発行所　株式会社エム・アイ・エー
〒150 東京都渋谷区渋谷2-9-1　青山田中ビル
電 話 (03)486-4500
編集制作　アスキー出版局第二書籍編集部
電 話 (03)486-4512

印刷・製本　東京音楽図書株式会社
ISBN4-87170-031-3　C3055 ¥2,500E